

Robust Supervisors for Intersection Collision Avoidance in the Presence of Uncontrolled Vehicles

Journal Title
XX(X):1–21
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/



Heejin Ahn¹, Andrea Rizzi², Alessandro Colombo³, and Domitilla Del Vecchio¹

Abstract

We present the design and validation of a centralized controller, called a *supervisor*, for collision avoidance of multiple human-driven vehicles at a road intersection, considering measurement errors, unmodeled dynamics, and uncontrolled vehicles. We design the supervisor to be least restrictive, that is, to minimize its interferences with human drivers. This performance metric is given a precise mathematical form by splitting the design process into two subproblems: verification problem and supervisor-design problem. The verification problem determines whether an input signal exists that makes controlled vehicles avoid collisions at all future times. The supervisor is designed such that if the verification problem returns yes, it allows the drivers' desired inputs; otherwise, it overrides controlled vehicles to prevent collisions. As a result, we propose *exact* and *efficient* supervisors. The exact supervisor solves the verification problem exactly but with combinatorial complexity. In contrast, the efficient supervisor solves the verification problem within a quantified approximation bound in polynomially bounded time with the number of controlled vehicles. We validate the performances of both supervisors through simulation and experimental testing.

1 Introduction

Autonomous robots have drawn attention in various applications, such as exploring unknown environment (Maimone et al. (2007)), moving materials in warehouses (Wurman et al. (2008)), and collecting data on ocean conditions (Smith et al. (2010)). Recently, there has been extensive research on autonomous vehicles from academic, industrial, and governmental sectors for the purpose of reducing the number of traffic accidents. Research has focused on developing fully autonomous vehicles as well as improving safety of human-driven vehicles by means of newly available automation, sensing, and communication capabilities. A major obstacle to the development of collision avoidance architecture for large traffic networks is computational complexity.

In general, there have been several approaches to reduce computational complexity in collision avoidance for multiple vehicles. In a decentralized framework, each vehicle makes its own decision to avoid collisions with its neighboring vehicles, thereby dividing a large problem into smaller local problems. To design a decentralized control law, Hoffmann and Tomlin (2008) and Gillula et al. (2011) used reachability analysis for hybrid systems, and Mastellone et al. (2008) defined a potential function. While computationally efficient, this framework is usually more conservative and unable to prevent a deadlock, where no further

control input exists to terminate processes (Cassandras and Lafortune (2008)). A centralized framework considers a whole system and thus can be less conservative, but computationally demanding. Collision avoidance problems were formulated into mixed integer linear programming (MILP) by assuming discrete-time linear vehicle dynamic models (Richards et al. (2002); Borrelli et al. (2006)) and considering geometric construction of collisions with vehicle models of instantaneous speed or angle changes (Pallottino et al. (2002); Alonso-Ayuso et al. (2011)). These MILP formulations are then solved by commercially available software. In this paper, we present a centralized controller, in which the collision avoidance problem is translated into a scheduling problem and then solved by solving this

¹Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

²Tri-Institutional Program in Computational Biology and Medicine, Weill Cornell Medical College, New York, NY, USA

³Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

Corresponding author:

Heejin Ahn, Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA.

Email: hjahn@mit.edu

scheduling problem. Moreover, we provide an approximate solution of this problem.

In collision avoidance problems at road intersections, the complexity can be mitigated to some extent by exploiting the fact that vehicles follow predefined paths, and side-impacts can be avoided by approximately scheduling their time of occupancy of the shared intersection (Peng and Akella (2005a,b)). Based on this concept, several autonomous intersection management schemes have been studied. Kowshik et al. (2011) proposed a hybrid architecture comprising an interplay between centralized coordination and distributed agents. Centralized coordination assigns time slots to agents, and distributed agents determine if they can cross an intersection within allocated time slots while avoiding rear-end collisions. Wu et al. (2012) employed an ant colony algorithm as an approximate solution for finding an optimal sequences of vehicles to improve traffic efficiency. Collision avoidance problems were formulated into nonlinear constrained optimization to eliminate overlaps of given trajectories inside an intersection (Lee and Park (2012)) and to minimize the risk of collision (Kamal et al. (2014)). These works consider fully autonomous vehicles and can solve collision avoidance problems by finding one safe input. However, when human operators drive vehicles, a controller needs to be least-restrictive, that is, override human drivers only when they can cause a collision.

To ensure least restrictiveness, all possible inputs must be taken into account, usually at expense of computational cost. Moreover, this exhaustive evaluation must be done frequently because controllers must keep monitoring vehicles' safety and intervene only when drivers are unable to prevent collisions. In Hafner and Del Vecchio (2011); Hafner et al. (2013), a safety control was designed for two vehicles. Their controller was validated in laboratory experiments (Hafner and Del Vecchio (2011)) and field experiments (Hafner et al. (2013)). Colombo and Del Vecchio (2012) translated a collision detection problem to a scheduling problem and employed a scheduling algorithm to design a safety control for multiple vehicles. Rear-end collisions as well as intersection collisions were considered in Colombo and Del Vecchio (2014).

In this paper, we propose a least-restrictive controller, called a *supervisor*, that prevents intersection collisions among human-driven vehicles. Our work extends the result of Colombo and Del Vecchio (2012) in that 1) we consider sources of uncertainty, including measurement errors, unmodeled dynamics, and uncontrolled vehicles, and 2) we perform a lab-based experiment to validate the supervisor in a setting subject to many sources of uncertainty. Here, controlled vehicles communicate with and are controlled by the supervisor, whereas uncontrolled vehicles are not. The inclusion of uncontrolled vehicles accounts for a realistic

mixed-traffic scenario where unequipped vehicles still travel on roads.

To design a supervisor, we formulate two problems: verification problem and supervisor-design problem. The verification problem determines the existence of an input signal that makes controlled vehicles avoid all future collisions. We prove that this problem is equivalent to an Inserted Idle-Time (IIT) scheduling problem, where an inserted idle-time represents a set of time intervals during which uncontrolled vehicles can occupy the intersection. This formulation was introduced in Ahn et al. (2014) to account for uncontrolled vehicles under perfect measurement and dynamic models. The supervisor is designed to override the drivers of controlled vehicles only when the verification problem determines that there will be no input signal to avoid collisions.

In order to study the trade-off between exactness and computational efficiency, we propose *exact* and *efficient* supervisors. The exact supervisor solves the verification problem exactly, and thus, overrides drivers only when strictly necessary. However, since the verification problem has combinatorial complexity, this supervisor is not scalable with the number of controlled vehicles. To improve the computational efficiency, the efficient supervisor is designed to solve the verification problem within a quantified approximation bound in polynomially bounded time. We validate the supervisors by performing computer simulations and lab-based experimental testing. Some of these experimental results were presented in Ahn et al. (2015).

This paper is organized as follows. In Section 2, we define an intersection model and a vehicle dynamic model. In Section 3, we formulate two problems: verification problem and supervisor-design problem, which are solved exactly in Section 4 and approximately in Section 5. The simulation results are given in Section 6, and the experimental results in Section 7.

2 System Definition

In this section, we introduce an intersection model and the vehicle dynamics.

2.1 Intersection model

Consider n human-driven vehicles approaching an intersection along n different paths. As shown in Figure 1, the intersection is modeled as an area containing the points at which these paths intersect. An open interval $(\alpha_i, \beta_i) \subset \mathbb{R}$ denotes the location of this conflict area on the longitudinal path of vehicle i . Among n vehicles, n_c vehicles are communicating with a supervisor, which takes control of these vehicles when potential crashes are detected and returns control back to the drivers when there

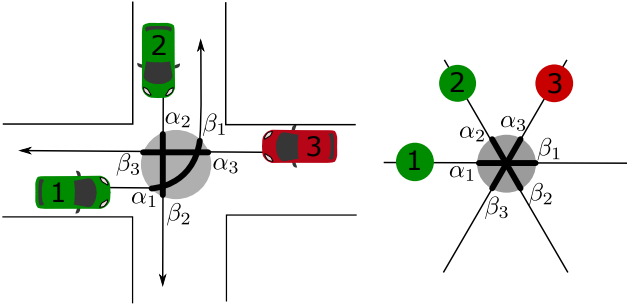


Figure 1. Road intersection (left) and its simplified model (right). We assume that the vehicles follow the prescribed paths, which intersect at one point. An intersection is modeled as an area containing this point (the shaded area), and its location is denoted by an open interval (α_i, β_i) along the path of vehicle i .

is no more threat. The other $n_{\bar{c}} = n - n_c$ vehicles are not communicating with, and can never be controlled by the supervisor. The supervisor however can measure their positions and speeds and use these measurements to predict their possible future behaviors. We define a controlled set \mathcal{C} as a set of controlled vehicles and an uncontrolled set $\bar{\mathcal{C}}$ as a set of uncontrolled vehicles. For notational simplicity, we number the vehicles such that $\mathcal{C} = \{1, 2, \dots, n_c\}$ and $\bar{\mathcal{C}} = \{n_c + 1, n_c + 2, \dots, n\}$.

2.2 Vehicle model

To describe the longitudinal dynamics of vehicle $i \in \mathcal{C} \cup \bar{\mathcal{C}}$, we introduce a state $x_i(t) = (y_i(t), v_i(t)) \in X_i \subset \mathbb{R}^2$, where $y_i(t) \in Y_i \subseteq \mathbb{R}$ and $v_i(t) \in [v_{i,min}, v_{i,max}] \subset \mathbb{R}$ are the position and the speed along the longitudinal path. Let $d_i(t) = (d_{y,i}(t), d_{v,i}(t)) \in \mathbb{R}^2$ denote a disturbance, where $d_{y,i}(t) \in [d_{y,i,min}, d_{y,i,max}]$ and $d_{v,i}(t) \in [d_{v,i,min}, d_{v,i,max}]$ account for unmodeled dynamics of $y_i(t)$ and $v_i(t)$, respectively. The longitudinal dynamics of controlled vehicle $j \in \mathcal{C}$ and uncontrolled vehicle $\gamma \in \bar{\mathcal{C}}$ are modeled as

$$\dot{x}_j = F_j(x_j, u_j, d_j), \quad \dot{x}_\gamma = F_\gamma(x_\gamma, w_\gamma, d_\gamma), \quad (1)$$

where $u_j \in U_j := [u_{j,min}, u_{j,max}] \subset \mathbb{R}$ is a throttle or brake input to controlled vehicle j applied by a supervisor, and $w_\gamma \in [w_{\gamma,min}, w_{\gamma,max}] \subset \mathbb{R}$ is a driver-input to uncontrolled vehicle γ .

The parallel composition of (1) is denoted as follows:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, \mathbf{w}, \mathbf{d}),$$

where $\mathbf{x} = (x_1, \dots, x_{n_c}, x_{n_c+1}, \dots, x_n) \in \mathbf{X}$, $\mathbf{u} = (u_1, u_2, \dots, u_{n_c}) \in \mathbf{U}$, $\mathbf{w} = (w_{n_c+1}, w_{n_c+2}, \dots, w_n)$, and $\mathbf{d} = (d_1, \dots, d_{n_c}, d_{n_c+1}, \dots, d_n)$. The output of the system is the position of all vehicles, denoted by $\mathbf{y} = (y_1, \dots, y_{n_c}, y_{n_c+1}, \dots, y_n) \in \mathbf{Y}$.

Let $u_j(\cdot) \in \mathcal{U}_j$ and $w_\gamma(\cdot) \in \mathcal{W}_\gamma$ denote an input signal to controlled vehicle $j \in \mathcal{C}$ and a driver-input signal to uncontrolled vehicle $\gamma \in \bar{\mathcal{C}}$, respectively. Let $d_i(\cdot) = (d_{y,i}(\cdot), d_{v,i}(\cdot)) \in \mathcal{D}_i$ denote a disturbance signal of vehicle $i \in \mathcal{C} \cup \bar{\mathcal{C}}$. We say $u_j(\cdot) \leq u'_j(\cdot)$ if $u_j(t) \leq u'_j(t)$ for all t . Similarly, $w_\gamma(\cdot) \leq w'_\gamma(\cdot)$ if $w_\gamma(t) \leq w'_\gamma(t)$ for all t , and $d_i(\cdot) \leq d'_i(\cdot)$ if $d_{y,i}(t) \leq d'_{y,i}(t)$ and $d_{v,i}(t) \leq d'_{v,i}(t)$ for all t .

Let $x_j(t, u_j(\cdot), d_j(\cdot), x_j(0))$ denote the state of controlled vehicle j at time t with an input signal $u_j(\cdot)$ and a disturbance signal $d_j(\cdot)$ starting at an initial state $x_j(0)$. Similarly, let $x_\gamma(t, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma(0))$ denote the state of uncontrolled vehicle γ at time t with a driver-input signal $w_\gamma(\cdot)$ and a disturbance signal $d_\gamma(\cdot)$ starting at $x_\gamma(0)$. Unless the input signals and initial state are important, we write $x_i(t)$ and $x_\gamma(t)$. We say $x_i(t) \leq x'_i(t)$ if $y_i(t) \leq y'_i(t)$ and $v_i(t) \leq v'_i(t)$ for all t for $i \in \mathcal{C} \cup \bar{\mathcal{C}}$ and make the following assumptions.

Assumption 1. The functions F_j and F_γ in (1) are order-preserving. That is, if $u_j(\cdot) \leq u'_j(\cdot)$, we have

$$x_j(t, u_j(\cdot), d_j(\cdot), x_j(0)) \leq x_j(t, u'_j(\cdot), d_j(\cdot), x_j(0)).$$

Similarly, if $d_j(\cdot) \leq d'_j(\cdot)$, we have

$$x_j(t, u_j(\cdot), d_j(\cdot), x_j(0)) \leq x_j(t, u_j(\cdot), d'_j(\cdot), x_j(0)).$$

Furthermore, if $x_j(0) \leq x'_j(0)$, we have

$$x_j(t, u_j(\cdot), d_j(\cdot), x_j(0)) \leq x_j(t, u_j(\cdot), d_j(\cdot), x'_j(0)).$$

The same relations hold for the state of uncontrolled vehicle γ .

Assumption 2. For all $i \in \mathcal{C} \cup \bar{\mathcal{C}}$, $\dot{y}_i(t) \geq 0$ for all t . Thus, the outputs $y_j(t, u_j(\cdot), d_j(\cdot), x_j(0))$ and $y_\gamma(t, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma(0))$ are non-decreasing in t .

Assumption 3. For all $j \in \mathcal{C}$, $x_j(t, u_j(\cdot), d_j(\cdot), x_j(0))$ is continuously dependent on $u_j(\cdot) \in \mathcal{U}_j$, and the input signal space \mathcal{U}_j is path-connected.

Let $\mathbf{x}(t, \mathbf{u}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(0))$ denote the aggregate state with $\mathbf{u}(\cdot) \in \mathbf{U}$, $\mathbf{w}(\cdot) \in \mathbf{W}$, and $\mathbf{d}(\cdot) \in \mathbf{D}$. This can also be written as $\mathbf{x}(t)$ if the other arguments are not important.

Notice that the vehicle dynamics (1) are subject to uncertainty originated from disturbances d_j and d_γ and an unknown driver-input w_γ . In addition to these, we consider a measurement noise $\delta_i := (\delta_{y,i}, \delta_{v,i})$ for $i \in \mathcal{C} \cup \bar{\mathcal{C}}$, where $\delta_{y,i} \in \mathbb{R}$ and $\delta_{v,i} \in \mathbb{R}$ are noises on the position measurement $y_{m,i}(t)$ and the speed measurement $v_{m,i}(t)$ at some time t , respectively. Then, the actual state $x_i(t) = (y_i(t), v_i(t))$ satisfies the following equation:

$$y_i(t) = y_{m,i}(t) + \delta_{y,i}, \quad v_i(t) = v_{m,i}(t) + \delta_{v,i}. \quad (2)$$

Let $x_{m,i}(t) = (y_{m,i}(t), v_{m,i}(t))$ denote the state measurement. Then, (2) can be rewritten as $x_i(t) = x_{m,i}(t) + \delta_i$. We make an assumption as follows.

Assumption 4. The measurement noise is bounded, that is, $\delta_i \in [\delta_{i,min}, \delta_{i,max}]$.

The aggregate state measurement is denoted by $\mathbf{x}_m(t)$, and then, $\mathbf{x}(t) = \mathbf{x}_m(t) + \delta$, where $\delta \in \Delta := [\delta_{min}, \delta_{max}]$ is the aggregated measurement noise.

2.3 The state estimation

We define a set of states $[\mathbf{x}^a(t), \mathbf{x}^b(t)]$, called the state estimation, that provides a lower and upper bound of the exact state, that is, $\mathbf{x}(t) \in [\mathbf{x}^a(t), \mathbf{x}^b(t)]$ for all t . At $t = 0$, the state estimation is defined as $\mathbf{x}^a(0) = \mathbf{x}_m(0) + \delta_{min}$ and $\mathbf{x}^b(0) = \mathbf{x}_m(0) + \delta_{max}$, so that $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$ because of (2) and Assumption 4. Given the initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$ and an input signal $\mathbf{u}(\cdot)$, the state estimation $[\mathbf{x}^a(t, \mathbf{u}(\cdot), \mathbf{x}^a(0)), \mathbf{x}^b(t, \mathbf{u}(\cdot), \mathbf{x}^b(0))]$ at time t is defined as follows: for $j \in \mathcal{C}$,

$$\begin{aligned} x_j^a(t, u_j(\cdot), x_j^a(0)) &:= \min_{\substack{d_j(\cdot) \in \mathcal{D}_j, x_j(0) \\ \in [x_j^a(0), x_j^b(0)]}} x_j(t, u_j(\cdot), d_j(\cdot), x_j(0)), \\ x_j^b(t, u_j(\cdot), x_j^b(0)) &:= \max_{\substack{d_j(\cdot) \in \mathcal{D}_j, x_j(0) \\ \in [x_j^a(0), x_j^b(0)]}} x_j(t, u_j(\cdot), d_j(\cdot), x_j(0)). \end{aligned} \quad (3)$$

For $\gamma \in \bar{\mathcal{C}}$,

$$\begin{aligned} x_\gamma^a(t) &:= \min_{\substack{w_\gamma(\cdot) \in \mathcal{W}_\gamma, d_\gamma(\cdot) \in \mathcal{D}_\gamma, \\ x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]}} x_\gamma(t, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma(0)), \\ x_\gamma^b(t) &:= \max_{\substack{w_\gamma(\cdot) \in \mathcal{W}_\gamma, d_\gamma(\cdot) \in \mathcal{D}_\gamma, \\ x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]}} x_\gamma(t, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma(0)). \end{aligned} \quad (4)$$

The state estimation is denoted by $[\mathbf{x}^a(t, \mathbf{u}(\cdot), \mathbf{x}^a(0)), \mathbf{x}^b(t, \mathbf{u}(\cdot), \mathbf{x}^b(0))]$ or $[\mathbf{x}^a(t), \mathbf{x}^b(t)]$ if the omitted arguments are not necessary.

By these definitions, the state estimation guarantees that given an initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$ and an input signal $\mathbf{u}(\cdot)$,

$$\begin{aligned} \mathbf{x}(t, \mathbf{u}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(0)) \\ \in [\mathbf{x}^a(t, \mathbf{u}(\cdot), \mathbf{x}^a(0)), \mathbf{x}^b(t, \mathbf{u}(\cdot), \mathbf{x}^b(0))], \end{aligned} \quad (5)$$

for all t for any $\mathbf{w}(\cdot) \in \mathcal{W}$ and $\mathbf{d}(\cdot) \in \mathcal{D}$ for any $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$.

Notice that $x_j^a(t, u_j(\cdot), x_j^a(0))$ and $x_j^b(t, u_j(\cdot), x_j^b(0))$ satisfy the inequalities in Assumption 1. We can define the position estimation, denoted by $[\mathbf{y}^a(t, \mathbf{u}(\cdot), \mathbf{x}^a(0)), \mathbf{y}^b(t, \mathbf{u}(\cdot), \mathbf{x}^b(0))]$ by letting $x_i^a(t) = (y_i^a(t), \cdot)$ and $x_i^b(t) = (y_i^b(t), \cdot)$ for all $i \in \mathcal{C} \cup \bar{\mathcal{C}}$. Throughout this paper, we use ‘ \cdot ’ in a vector if specifying the entry is not important. The position estimation also inherits the order-preserving property from $\mathbf{y}(t)$. Later in Section 4.2, this estimation can be updated using the state measurement.

3 Problem Statement

Let us define an intersection collision. Since an intersection is modeled as a single conflict area, we consider a collision occurs if at least two vehicles are simultaneously inside the intersection. The output configurations corresponding to collisions belong to the *Bad set*, denoted by $B \subset Y$. The Bad set is defined as follows:

$$\begin{aligned} B &:= \{\mathbf{y} \in Y : y_i \in (\alpha_i, \beta_i) \text{ and } y_j \in (\alpha_j, \beta_j) \\ &\text{for some } i \neq j \text{ such that } i \in \mathcal{C} \cup \bar{\mathcal{C}} \text{ and } j \in \mathcal{C}\}. \end{aligned}$$

Throughout this paper, we assume that uncontrolled vehicles do not crash among themselves. This assumption enables us to focus on preventing collisions in which at least one controlled vehicle is involved.

A supervisor runs in discrete time with a time step τ . At time $k\tau$ where k is a nonnegative integer, a desired input $\mathbf{u}_{desire}^k \in \mathbf{U}$ and a state $\mathbf{x}_m(k\tau) \in \mathbf{X}$ are measured. Then, a state estimation $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$ is updated to satisfy $\mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)] \subseteq [\mathbf{x}_m(k\tau) + \delta_{min}, \mathbf{x}_m(k\tau) + \delta_{max}]$. The desired input \mathbf{u}_{desire}^k is a vector of current inputs of controlled vehicles’ drivers at time $k\tau$. As illustrated in Figure 2, we define desired input signals $\mathbf{u}_{desire}^k(\cdot) \in \mathcal{U}$ on time $[k\tau, (k+1)\tau)$ and $\mathbf{u}_{desire}^{k,\infty}(\cdot) \in \mathcal{U}$ on time $[k\tau, \infty)$ such that

$$\mathbf{u}_{desire}^k(t) = \mathbf{u}_{desire}^{k,\infty}(t) = \mathbf{u}_{desire}^k \text{ for } t \in [k\tau, (k+1)\tau). \quad (6)$$

Also, we define a safe input signal $\mathbf{u}_{safe}^{k,\infty}(\cdot)$ on time $[k\tau, \infty)$ such that

$$\begin{aligned} \forall \mathbf{w}(\cdot) \in \mathcal{W}, \mathbf{d}(\cdot) \in \mathcal{D}, \mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \\ \mathbf{y}(t, \mathbf{u}_{safe}^{k,\infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B \text{ for all } t. \end{aligned} \quad (7)$$

Let $\mathbf{u}_{safe}^k(\cdot)$ be $\mathbf{u}_{safe}^{k,\infty}(\cdot)$ restricted to time $[k\tau, (k+1)\tau)$.

The supervisor $s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k)$ is designed according to the following problem, which is illustrated in Figure 3.

Problem 1. (Supervisor-design). Design a supervisor $s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k)$ such that for any $\mathbf{w}(\cdot), \mathbf{d}(\cdot)$, and $\mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$, it returns

$$\begin{cases} \mathbf{u}_{desire}^k(\cdot) & \text{if } \exists \mathbf{u}_{desire}^{k,\infty}(\cdot) : \text{for all } t \geq 0 \\ & \mathbf{y}(t, \mathbf{u}_{desire}^{k,\infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B \\ \mathbf{u}_{safe}^k(\cdot) & \text{otherwise,} \end{cases}$$

and it is non-blocking, that is, if $s([\mathbf{x}^a((k-1)\tau), \mathbf{x}^b((k-1)\tau)], \mathbf{u}_{desire}^{k-1}) \neq \emptyset$, then for any $\mathbf{u}_{desire}^k \in \mathbf{U}$, we have $s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k) \neq \emptyset$.

The supervisor in Problem 1 is least restrictive in the sense that overrides are activated only when the desired input of controlled vehicles’ drivers would lead to collisions at some

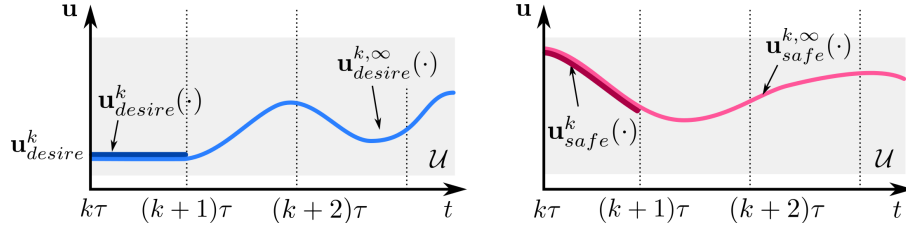


Figure 2. $u_{desire}^{k,\infty}(\cdot)$ is an artificially constructed signal that equals to u_{desire}^k for time $[k\tau, (k+1)\tau]$ as in (6). $u_{safe}^{k,\infty}(\cdot)$ is a safe input signal that makes the system avoid the Bad set as in (7).

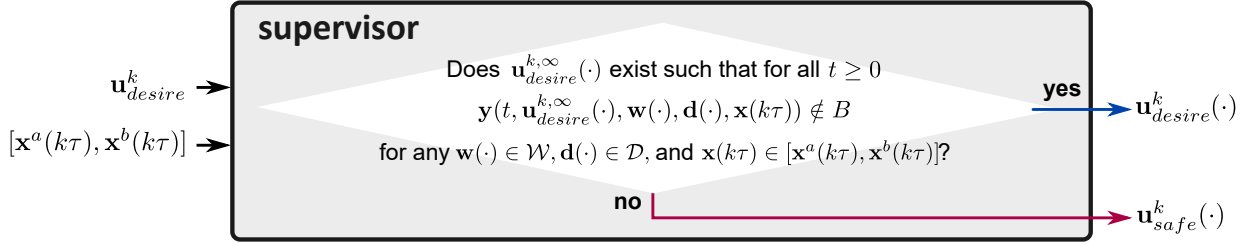


Figure 3. Illustration of the supervisor. At time $k\tau$, the supervisor takes the desired input of controlled vehicles' drivers u_{desire}^k and the state estimation $[x^a(k\tau), x^b(k\tau)]$. Depending on the existence of $u_{desire}^{k,\infty}(\cdot)$ that prevents all the vehicles from entering the Bad set for all t for any $w(\cdot) \in \mathcal{W}$, $d(\cdot) \in \mathcal{D}$, and $x(k\tau) \in [x^a(k\tau), x^b(k\tau)]$, the supervisor either allows the drivers to travel with their desired input $u_{desire}^k(\cdot)$ or overrides them with the safe input $u_{safe}^k(\cdot)$.

future times. To distinguish between safe and unsafe inputs, the supervisor needs to verify that the state reached using the desired input is compatible with a safe evolution of the system. This introduces the following problem.

Problem 2. (Verification). Given an initial state estimation $[x^a(0), x^b(0)]$, determine if there exists an input signal $u(\cdot)$ that guarantees $y(t, u(\cdot), w(\cdot), d(\cdot), x(0)) \notin B$ for all $t \geq 0$ for any $w(\cdot) \in \mathcal{W}$, $d(\cdot) \in \mathcal{D}$, and $x(0) \in [x^a(0), x^b(0)]$.

By solving Problem 2, the supervisor determines if an override is necessary. If the desired input leads to a set of states at which Problem 2 returns *no*, the supervisor overrides controlled vehicles with a safe input signal. The two above problems are solved exactly in the next section, and approximately in Section 5.

4 Exact solutions

In this section, we provide the solutions of the verification problem (Problem 2) and the supervisor-design problem (Problem 1). To solve Problem 2, we formulate an Inserted Idle-Time scheduling problem, which is solved straightforwardly, and prove that this problem is equivalent to Problem 2. We then propose a solution to Problem 1.

4.1 Inserted Idle-Time scheduling problem

Problem 2 can be translated into a scheduling problem, considering the intersection as a resource that all vehicles must share. The schedule represents a sequence of the times

at which each controlled vehicle enters an intersection, and the idle-times represent the sets of times during which an intersection is potentially occupied by uncontrolled vehicles. This analogy is characterized mathematically using the concept of decision problem equivalence.

Definition 1. (Cormen et al. (2009)) Consider two decision problems A and B . The input to a particular problem is called an **instance** of that problem. Then, A is reducible to B if there is a procedure that transforms any instance α of A into some instance β of B in polynomial time, and the answer for α is “yes”, denoted by $\alpha \in A$, if and only if the answer for β is “yes”, denoted by $\beta \in B$. We say A is **equivalent** to B if and only if A is reducible to B and B is reducible to A .

An instance I of Problem 2 is described by $([x^a(0), x^b(0)], S)$ where $S := (\mathbf{F}, \mathbf{X}, \mathbf{Y}, \mathcal{U}, \mathcal{W}, \mathcal{D}, \mathcal{C}, \bar{\mathcal{C}}, \alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n)$.

To formulate the IIT scheduling problem, we define scheduling parameters.

Definition 2. Given an initial state estimation $[x^a(0), x^b(0)]$, release times R_j , deadlines D_j , and process times $P_j(T_j)$ are defined for controlled vehicles. For $j \in \mathcal{C}$, if $y_j^b(0) < \alpha_j$,

$$R_j := \min_{u_j(\cdot) \in \mathcal{U}_j} \{t \geq 0 : y_j^b(t, u_j(\cdot), x_j^b(0)) = \alpha_j\},$$

$$D_j := \max_{u_j(\cdot) \in \mathcal{U}_j} \{t \geq 0 : y_j^b(t, u_j(\cdot), x_j^b(0)) = \alpha_j\}.$$

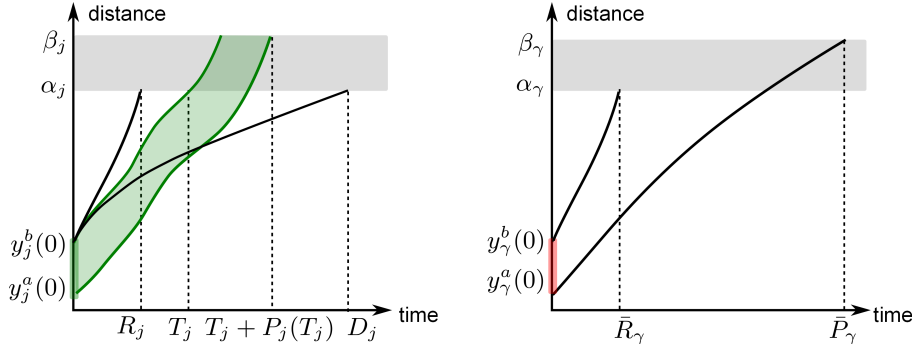


Figure 4. Scheduling parameters in Definition 2. Referring to Figure 1, (α_j, β_j) is the location of an intersection along the route of vehicle j .

Given a real number $T_j > 0$,

$$P_j(T_j) := \min_{u_j(\cdot) \in \mathcal{U}_j} \{t - T_j \geq 0 : \\ y_i^a(t, u_j(\cdot), x_j^a(0)) = \beta_j \\ \text{with constraint } y_j^b(T_j, u_j(\cdot), x_j^b(0)) = \alpha_j\}.$$

If $y_j^b(0) \geq \alpha_j$, then $R_j = 0, D_j = 0$, and $P_j(T_j) = \min_{u_j(\cdot) \in \mathcal{U}_j} \{t : y_j^a(t, u_j(\cdot), x_j^a(0)) = \beta_j\}$. If $y_j^a(0) \geq \beta_j$, then set $R_j = 0, D_j = 0$, and $P_j(T_j) = 0$. If the constraint is not satisfied, set $P_j(T_j) = \infty$.

Idle-times $(\bar{R}_\gamma, \bar{P}_\gamma)$ are defined for uncontrolled vehicles. For $\gamma \in \bar{\mathcal{C}}$, if $y_\gamma^b(0) < \alpha_\gamma$,

$$\bar{R}_\gamma := \{t \geq 0 : y_\gamma^b(t) = \alpha_\gamma\}, \\ \bar{P}_\gamma := \{t \geq 0 : y_\gamma^a(t) = \beta_\gamma\}.$$

If $y_\gamma^b(0) \geq \alpha_\gamma$, set $\bar{R}_\gamma = 0$ and $\bar{P}_\gamma = \{t : y_\gamma^a(t) = \beta_\gamma\}$. If $y_\gamma^a(0) \geq \beta_\gamma$, set $\bar{R}_\gamma = 0$ and $\bar{P}_\gamma = 0$.

In this definition, release time R_j is the earliest that controlled vehicle j can enter an intersection while deadline D_j is the latest. Given that controlled vehicle j enters the intersection no earlier than time T_j , process time $P_j(T_j)$ is the earliest that it can cross the intersection. Uncontrolled vehicle γ enters and exits the intersection within the idle-time $(\bar{R}_\gamma, \bar{P}_\gamma)$ regardless of its driver-input and disturbance signals. These parameters are illustrated in Figure 4.

The Inserted Idle-time (IIT) scheduling problem is formulated as follows.

Problem 3. (IIT scheduling). Given an initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$, determine whether there exists a schedule $\mathbf{T} = (T_1, \dots, T_{n_c}) \in \mathbb{R}_+^{n_c}$ such that for all $j \in \mathcal{C}$,

$$R_j \leq T_j \leq D_j, \quad (8)$$

for all $i \neq j \in \mathcal{C}$,

$$(T_i, T_i + P_i(T_i)) \cap (T_j, T_j + P_j(T_j)) = \emptyset, \quad (9)$$

for all $j \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$,

$$(T_j, T_j + P_j(T_j)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset. \quad (10)$$

Notice that T_j and $P_j(T_j)$ are defined such that for some $u_j(\cdot)$, $y_j^b(T_j, u_j(\cdot), x_j^b(0)) = \alpha_j$ and $y_j^a(T_j + P_j(T_j), u_j(\cdot), x_j^a(0)) = \beta_j$. Condition (8) represents the constraint induced on the schedule by the bounded input signals. By condition (9), the times during which controlled vehicles i and j occupy the intersection for any disturbance do not overlap. This implies that a collision between controlled vehicles i and j is averted. Similarly, condition (10) implies that vehicle j does not occupy the intersection during the idle-time, thereby preventing a collision between controlled vehicle j and uncontrolled vehicle γ . Thus, a schedule satisfying the above conditions is related to an input signal that can prevent any future collision. This is the essence of the proof of the following theorem.

Theorem 1. Problem 2 and Problem 3 are equivalent.

Proof. By Definition 1, we need to show two things: Problem 2 is reducible to Problem 3, and Problem 3 is reducible to Problem 2. Notice that an instance I of Problem 3 is also described by $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S)$, which is the same as an instance of Problem 2. Thus, the transformation between the instances of these problems takes constant time. Now, the following relation is left to prove the equivalence.

$$I \in \text{Problem 2} \Leftrightarrow I \in \text{Problem 3}.$$

(\Rightarrow) Given an initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$, there is an input signal $\tilde{\mathbf{u}}(\cdot) \in \mathcal{U}$ such that $\mathbf{y}(t, \tilde{\mathbf{u}}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(0)) \notin B$ for all t for any $\mathbf{w}(\cdot) \in \mathcal{W}$ and $\mathbf{d}(\cdot) \in \mathcal{D}$ for any $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$.

Using $\tilde{u}_j(\cdot)$ for $j \in \mathcal{C}$, which denotes the j -th element of $\tilde{\mathbf{u}}(\cdot)$, let \tilde{T}_j be the time at which $y_j^b(t, \tilde{u}_j(\cdot), x_j^b(0)) = \alpha_j$ if $y_j^b(0) < \alpha_j$. If $y_j^b(0) \geq \alpha_j$, set $\tilde{T}_j = 0$. This satisfies the constraint of $P_j(\tilde{T}_j)$ in Definition 2. Let $\tilde{T}_j + \tilde{P}_j(\tilde{T}_j)$ be the

time at which $y_j^a(t, \tilde{u}_j(\cdot), x_j^a(0)) = \beta_j$. Set $\tilde{P}_j(\tilde{T}_j) = 0$ if $y_j^a(0) \geq \beta_j$.

By the definitions of R_j and D_j , condition (8) is satisfied. Suppose without loss of generality, $\tilde{T}_i \leq \tilde{T}_j$. At $t = \tilde{T}_j$, we have $y_j^b(t, \tilde{u}_j(\cdot), x_j^b(0)) = \alpha_j$. Since $\tilde{u}_i(\cdot)$ and $\tilde{u}_j(\cdot)$ guarantee that at most one vehicle is inside an intersection, we must have $y_i^a(t, \tilde{u}_i(\cdot), x_i^a(0)) \geq \beta_i$. Because $y_j^a(t)$ is non-decreasing in time, $\tilde{T}_i + \tilde{P}_i(\tilde{T}_i) \leq t$. By the definition of $P_i(\tilde{T}_i)$, we have $\tilde{T}_i + P_i(\tilde{T}_i) \leq \tilde{T}_i + \tilde{P}_i(\tilde{T}_i) \leq t = \tilde{T}_j$, which concludes $(\tilde{T}_i, \tilde{T}_i + P_i(\tilde{T}_i)) \cap (\tilde{T}_j, \tilde{T}_j + P_j(\tilde{T}_j)) = \emptyset$ (condition (9)).

In order to avoid the Bad set, vehicle $\gamma \in \bar{\mathcal{C}}$ is not inside an intersection during $[\tilde{T}_j, \tilde{T}_j + \tilde{P}(\tilde{T}_j)]$ for any $w_\gamma(\cdot) \in \mathcal{W}_\gamma$, $d_\gamma(\cdot) \in \mathcal{D}_\gamma$, and $x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]$. Since $y_\gamma^b(\bar{R}_\gamma) = \alpha_\gamma$ and $y_\gamma^a(\bar{P}_\gamma) = \beta_\gamma$ by Definition 2, we have $(\tilde{T}_j, \tilde{T}_j + \tilde{P}_j(\tilde{T}_j)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. By the definition of $P_j(\tilde{T}_j)$, we have $(\tilde{T}_j, \tilde{T}_j + P_j(\tilde{T}_j)) \subset (\tilde{T}_j, \tilde{T}_j + \tilde{P}_j(\tilde{T}_j))$. Thus, $(\tilde{T}_j, \tilde{T}_j + P_j(\tilde{T}_j)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. (condition (10)).

This proves that there exists the schedule $\tilde{\mathbf{T}}$ that satisfies conditions (8), (9), and (10).

(\Leftarrow) Given an initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$, there exists a schedule $\tilde{\mathbf{T}} \in \mathbb{R}^{n_c}$ that satisfies the conditions of Problem 3.

For $j \in \mathcal{C}$, define $\tilde{u}_j(\cdot)$ such that $y_j^b(\tilde{T}_j, \tilde{u}_j(\cdot), x_j^b(0)) = \alpha_j$ if $y_j^b(0) < \alpha_j$. If $y_j^b(0) \geq \alpha_j$ and $y_j^a(0) \leq \beta_j$, define $\tilde{u}_j(\cdot)$ such that $y_j^a(\tilde{T}_j + P_j(\tilde{T}_j), \tilde{u}_j(\cdot), x_j^a(0)) = \beta_j$. If $y_j^a(0) > \beta_j$, we do not consider vehicle j because it has already crossed the intersection. Since $y_j^a(t, u_j(\cdot), x_j^a(0))$ depends continuously on $u_j(\cdot) \in \mathcal{U}_j$ and \mathcal{U}_j is path connected by Assumption 3, condition (8) implies that such an input signal exists, i.e., $\tilde{u}_j(\cdot) \in \mathcal{U}_j$.

Consider $\tilde{u}_i(\cdot)$ and $\tilde{u}_j(\cdot)$ for $i \neq j \in \mathcal{C}$ where $\tilde{T}_i \leq \tilde{T}_j$. Condition (9) says $\tilde{T}_i + P_i(\tilde{T}_i) \leq \tilde{T}_j$. At time $t \in [\tilde{T}_i + P_i(\tilde{T}_i), \tilde{T}_j]$, we have $y_i^a(t, \tilde{u}_i(\cdot), x_i^a(0)) \geq \beta_i$ and $y_j^b(t, \tilde{u}_j(\cdot), x_j^b(0)) \leq \alpha_j$. Thus, for any disturbance and initial condition, vehicle j enters the intersection after vehicle i leaves it.

For $j \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$, condition (10) implies either $\bar{P}_\gamma \leq T_j$ or $\tilde{T}_j + P_j(\tilde{T}_j) \leq \bar{R}_\gamma$. In the first case, at any time $t \in [\bar{P}_\gamma, T_j]$, we have $y_\gamma^a(t) \geq \beta_\gamma$ and $y_j^b(t, \tilde{u}_j(\cdot), x_j^b(0)) \leq \alpha_j$ so that for any $w_\gamma(\cdot) \in \mathcal{W}_\gamma$, $d_\gamma(\cdot) \in \mathcal{D}_\gamma$, $x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]$ and for any $d_j(\cdot) \in \mathcal{D}_j$, $x_j(0) \in [x_j^a(0), x_j^b(0)]$, after vehicle γ exits the intersection, vehicle j enters it. In the second case, at any time $t \in [\tilde{T}_j + P_j(\tilde{T}_j), \bar{R}_\gamma]$, we have $y_j^a(t, \tilde{u}_j(\cdot), x_j^a(0)) \geq \beta_j$ and $y_\gamma^b(t) \leq \alpha_\gamma$. This means after vehicle j exits the intersection for any $d_j(\cdot) \in \mathcal{D}_j$ and $x_j(0) \in [x_j^a(0), x_j^b(0)]$, vehicle γ enters it for any $w_\gamma(\cdot) \in \mathcal{W}_\gamma$, $d_\gamma(\cdot) \in \mathcal{D}_\gamma$, and $x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]$.

Therefore, the schedule $\tilde{\mathbf{T}}$ ensures that there exists an input signal $\tilde{\mathbf{u}}(\cdot)$ which prevents entering the Bad set. \square

We now provide an algorithm that solves Problem 3, which, in turn, solves Problem 2 by Theorem 1. This algorithm contains two procedures. The first procedure, SCHEDULING, generates a schedule \mathbf{T} given a sequence π such that $T_{\pi_i} \leq T_{\pi_j}$ if $i < j$, and evaluates whether \mathbf{T} satisfies conditions (8)-(10). Here, π_i denotes the i -th entry of π and T_{π_i} the π_i -entry of \mathbf{T} . The second procedure, EXACT, inspects all possible sequences until a sequence with a feasible schedule is found. If a feasible schedule is found, the answer of Problem 2 is yes, and otherwise, the answer is no.

Algorithm 1 focuses on vehicles before an intersection. The set of such vehicles is denoted by $\mathcal{M} := \{j \in \mathcal{C} : y_j^b(0) < \alpha_j\}$. For simplicity, let $y_j^b(0) < \alpha_j$ for $j \in \{1, \dots, |\mathcal{M}|\}$ and $y_j^b(0) \geq \alpha_j$ for $j \in \{|\mathcal{M}| + 1, \dots, n_c\}$. Let \mathcal{P} be a set of all permutations of \mathcal{M} . Its cardinality is $(|\mathcal{M}|)! = 1 \times 2 \times \dots \times |\mathcal{M}|$. Let $\pi \in \mathbb{R}^{|\mathcal{M}|}$ be a vector in \mathcal{P} . Let $\bar{\pi} \in \mathbb{R}^{n_c}$ be a vector of $\gamma \in \bar{\mathcal{C}}$ in an increasing order of \bar{R}_γ , that is, $\bar{R}_{\bar{\pi}_j} \leq \bar{R}_{\bar{\pi}_{j'}}$ for $j \leq j'$. Let $\pi_{0,j}$ and $\bar{\pi}_j$ denote the j -th entry of π_0 and $\bar{\pi}$, respectively.

Algorithm 1 Exact Solution of Problem 3

```

1: procedure SCHEDULING( $\pi_0, [\mathbf{x}^a(0), \mathbf{x}^b(0)], S$ )
2:   for all  $j \in \mathcal{C}$  do calculate  $R_j, D_j$ 
3:   for all  $\gamma \in \bar{\mathcal{C}}$  do calculate  $\bar{R}_\gamma, \bar{P}_\gamma$ 
4:    $\mathcal{M} = \{j \in \mathcal{C} : y_j^b(0) < \alpha_j\}$  and  $\bar{\mathcal{M}} = \mathcal{C} \setminus \mathcal{M}$ 
5:    $T_j = 0 \forall j \in \mathcal{M}$  and  $P_{max} = \max_{j \in \mathcal{M}} P_j(T_j)$ 
6:    $\pi = (\pi_{0,j} : \pi_{0,j} \in \mathcal{M} \text{ for } j = 1 \text{ to } |\pi_0|)$ 
7:   for  $i = 1$  to  $|\mathcal{M}|$  do
8:     if  $i = 1$  then  $T_{\pi_1} = \max(R_{\pi_1}, P_{max})$ 
9:     else if  $i \geq 2$  then
10:        $T_{\pi_i} = \max(R_{\pi_i}, T_{\pi_{i-1}} + P_{\pi_{i-1}}(T_{\pi_{i-1}}))$ 
11:     for  $j = 1$  to  $n_c$  do
12:       if  $T_{\pi_i} \geq \bar{R}_{\bar{\pi}_j}$  then
13:          $T_{\pi_i} = \max(T_{\pi_i}, \bar{P}_{\bar{\pi}_j})$ 
14:       else if  $T_{\pi_i} + P_{\pi_i}(T_{\pi_i}) > \bar{R}_{\bar{\pi}_j}$  then
15:          $T_{\pi_i} = \bar{P}_{\bar{\pi}_j}$ 
16:   if  $T_j \leq D_j$  for all  $j \in \mathcal{C}$  then return ( $\mathbf{T}, yes$ )
17:   return ( $\emptyset, no$ )
18: procedure EXACT( $[\mathbf{x}^a(0), \mathbf{x}^b(0)], S$ )
19:   if  $[\mathbf{y}^a(0), \mathbf{y}^b(0)] \cap B \neq \emptyset$  then return ( $\emptyset, no$ )
20:    $\mathcal{M} = \{j \in \mathcal{C} : y_j^b(0) < \alpha_j\}$  and  $\bar{\mathcal{M}} = \mathcal{C} \setminus \mathcal{M}$ 
21:   if  $|\mathcal{M}| = 0$  then return ( $\emptyset, yes$ )
22:   for all  $\pi \in \mathcal{P}$  do
23:     ( $\mathbf{T}, ans$ ) = SCHEDULING( $\pi, [\mathbf{x}^a(0), \mathbf{x}^b(0)], S$ )
24:     if  $ans = yes$  then return ( $\mathbf{T}, yes$ )
25:   return ( $\emptyset, no$ )

```

Procedure SCHEDULING in Algorithm 1 works as follows. In lines 2-3, the scheduling parameters in Definition 2 are

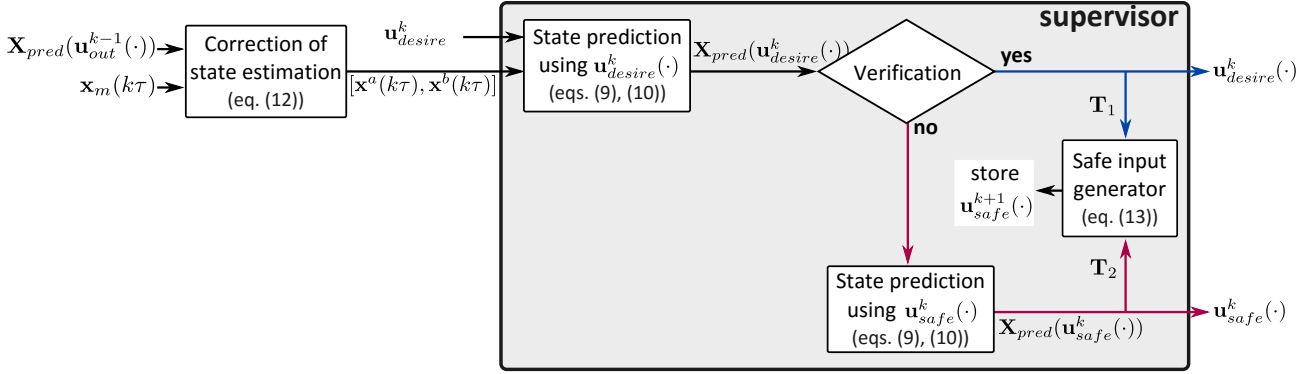


Figure 5. Schematic of the supervisor at time $k\tau$. Using the state measurement $\mathbf{x}_m(k\tau)$ and the previous step's state prediction $\mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))$, the state estimation is corrected to $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$. The supervisor takes $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$ and the current desired input \mathbf{u}_{desire}^k and returns either $\mathbf{u}_{desire}^k(\cdot)$ or $\mathbf{u}_{safe}^k(\cdot)$ depending on the answer of the verification problem. When the answer is *no*, the state prediction $\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot))$ guarantees the existence of a feasible schedule \mathbf{T}_2 , thereby ensuring the non-blocking property of the supervisor. At each step, a safe input signal $\mathbf{u}_{safe}^{k+1}(\cdot)$ is generated using a feasible schedule and stored for a possible use at the next step.

computed given an initial state estimation. In lines 4 and 5, if $y_j^b(0) \geq \alpha_j$, $T_j = 0$ because $R_j = D_j = 0$ and P_{max} is the time at which all vehicles $j \in \bar{\mathcal{M}}$ exit the intersection. The input argument π_0 is the vector of the indexes of all vehicles controlled by the supervisor. The assignment at line 6 extracts from π_0 the sub-vector π of vehicles which have not yet reached the intersection. The schedule needs only be computed for the sub-vector π .

In a given sequence π , vehicle π_{i-1} crosses the intersection earlier than vehicle π_i . Given this sequence π , procedure SCHEDULING finds the earliest possible schedule. In the **for** loop of lines 11-15, uncontrolled vehicle $\bar{\pi}_j$ is considered. If line 12 is true, then line 13 ensures that $T_{\pi_i} \geq \bar{P}_{\bar{\pi}_j}$ so that condition (10) is satisfied. Otherwise, if line 14 is true, meaning that $T_{\pi_i} < \bar{R}_{\bar{\pi}_j} < T_{\pi_i} + P_{\pi_i}(T_{\pi_i})$, then T_{π_i} is delayed to $\bar{P}_{\bar{\pi}_j}$ in line 15 so that condition (10) becomes satisfied. The schedule T_{π_i} takes $\bar{P}_{\bar{\pi}_j}$, which is the earliest possible value. Otherwise, $T_{\pi_i} + P_{\pi_i}(T_{\pi_i}) \leq \bar{R}_{\bar{\pi}_j}$ so that condition (10) is guaranteed. Lastly, in line 16, the right inequality of condition (8) is checked. If line 16 is true, this procedure returns a feasible schedule and answer *yes*. If $T_j > D_j$ for some j , since the schedule \mathbf{T} is constructed so that it is the earliest possible value, there cannot be another schedule $\bar{\mathbf{T}}$ such that $\bar{T}_j \leq D_j$. Thus, no feasible schedule can be found in this sequence π , and the procedure returns *no*.

Procedure EXACT in Algorithm 1 solves Problem 3 by inspecting all permutations in \mathcal{P} until a feasible schedule is found as noted in lines 22-24. In line 19, if a given initial position estimation is already inside the Bad set, the procedure returns *no*. In line 21, if $y_j^b(0) \geq \alpha_j$ for all $j \in \mathcal{C}$, then it returns *yes* since all controlled vehicles have entered the intersection. If no feasible schedule is found after

evaluating all permutations in \mathcal{P} , an empty set and answer *no* are returned in line 25.

The running time of procedure EXACT is determined by the **for** loops of lines 22-24. For the worst case, all permutations in \mathcal{P} must be evaluated. As the number of controlled vehicles increases, the computation time increases exponentially. Indeed, a scheduling problem is known to be NP-hard (Colombo and Del Vecchio (2012)). To avoid this computational complexity issue, we devise an approximate algorithm to solve the IIT scheduling problem with the guarantee of a quantified approximation bound. This approximate algorithm is provided in Section 5.1.

4.2 Exact supervisor

In this section, we solve Problem 1 by providing an algorithm implementing the exact supervisor.

The schematic of a supervisor at time $k\tau$ is illustrated in Figure 5. The state prediction computed at the previous step $\mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))$ is known, where $\mathbf{u}_{out}^{k-1}(\cdot)$ is the output that the supervisor returns at the previous step, that is, controlled vehicles traveled with $\mathbf{u}_{out}^{k-1}(\cdot)$ for time $[(k-1)\tau, k\tau)$. Also, the state measurement $\mathbf{x}_m(k\tau)$ is received. These state prediction and measurement are used to update the state estimation $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$. This state estimation and the desired input \mathbf{u}_{desire}^k are the inputs to the supervisor and used to predict the state at the next time step, denoted by $\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot))$. Given this state prediction, the verification problem is solved. If the answer is *yes*, the supervisor allows the controlled vehicles to travel with the desired input for time $[k\tau, (k+1)\tau)$. A safe input signal $\mathbf{u}_{safe}^{k+1}(\cdot)$ is generated using a feasible schedule \mathbf{T}_1 and stored for a possible use at the next time step. If the answer is *no*, a safe input signal $\mathbf{u}_{safe}^k(\cdot)$

stored at the previous step is used to override the controlled vehicles. It will be proved in this section that the state prediction $\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot))$ always has a feasible schedule \mathbf{T}_2 , that is, $\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)) \in \text{Problem 2}$. A safe input signal $\mathbf{u}_{safe}^{k+1}(\cdot)$ is generated using \mathbf{T}_2 and stored. The output $\mathbf{u}_{out}^k(\cdot)$ is either $\mathbf{u}_{desire}^k(\cdot)$ or $\mathbf{u}_{safe}^k(\cdot)$, and the state prediction $\mathbf{X}_{pred}(\mathbf{u}_{out}^k(\cdot))$ will be used at the next step to correct the state estimation with a new state measurement.

In the last section, we presented Algorithm 1 that solves the verification problem. This section describes the remaining components: state prediction, correction of the state estimation, and a safe input generator. Also, an algorithm to implement the supervisor is presented with the proof that this supervisor is the solution of Problem 1.

4.2.1 State prediction. Using the dynamic model (1), this function predicts the state reached at the next time step with a given input signal starting from a set of states. Suppose that at time $k\tau$, we have the state estimation $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$ and the input $\mathbf{u}^k(\cdot) \in \mathcal{U}$ defined on time $[k\tau, (k+1)\tau)$. The state prediction $\mathbf{X}_{pred}(\mathbf{u}^k(\cdot)) = [\min \mathbf{X}_{pred}(\mathbf{u}^k(\cdot)), \max \mathbf{X}_{pred}(\mathbf{u}^k(\cdot))] \subset \mathbf{X}$ is a set of possible states at the next time step. Its i -th set is denoted by $X_{pred,i}(\mathbf{u}_i^k(\cdot)) = [\min X_{pred,i}(\mathbf{u}_i^k(\cdot)), \max X_{pred,i}(\mathbf{u}_i^k(\cdot))]$. The state prediction is defined as follows: for $j \in \mathcal{C}$,

$$\begin{aligned} \min X_{pred,j}(\mathbf{u}_j^k(\cdot)) &= \min_{d_j(\cdot) \in \mathcal{D}_j} x_j(\tau, \mathbf{u}_j^k(\cdot), d_j(\cdot), x_j^a(k\tau)), \\ \max X_{pred,j}(\mathbf{u}_j^k(\cdot)) &= \max_{d_j(\cdot) \in \mathcal{D}_j} x_j(\tau, \mathbf{u}_j^k(\cdot), d_j(\cdot), x_j^b(k\tau)). \end{aligned} \quad (11)$$

For $\gamma \in \bar{\mathcal{C}}$,

$$\begin{aligned} \min X_{pred,\gamma} &= \min_{\substack{w_\gamma(\cdot) \in \mathcal{W}_\gamma, \\ d_\gamma(\cdot) \in \mathcal{D}_\gamma}} x_\gamma(\tau, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma^a(k\tau)), \\ \max X_{pred,\gamma} &= \max_{\substack{w_\gamma(\cdot) \in \mathcal{W}_\gamma, \\ d_\gamma(\cdot) \in \mathcal{D}_\gamma}} x_\gamma(\tau, w_\gamma(\cdot), d_\gamma(\cdot), x_\gamma^b(k\tau)). \end{aligned} \quad (12)$$

By definition, $\min \mathbf{X}_{pred}(\mathbf{u}^k(\cdot))$ is the smallest state propagated from $\mathbf{x}^a(k\tau)$ for time τ with the input $\mathbf{u}^k(\cdot)$ for any disturbance and any driver-input of uncontrolled vehicles, and $\max \mathbf{X}_{pred}(\mathbf{u}^k(\cdot))$ is the largest.

Notice that $\text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), S)$ in Algorithm 1 determines the existence of an input signal $\mathbf{u}_{safe}^{k+1,\infty}(\cdot)$ that satisfies $\mathbf{y}(t, \mathbf{u}_{safe}^{k+1,\infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}((k+1)\tau)) \notin B$ for all t for any $\mathbf{w}(\cdot)$ and $\mathbf{d}(\cdot)$ for any $\mathbf{x}((k+1)\tau) \in \mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot))$. If we define $\mathbf{u}_{desire}^{k,\infty}(\cdot)$ as

$$\mathbf{u}_{desire}^{k,\infty}(t) = \begin{cases} \mathbf{u}_{desire}^k & \text{for } t \in [k\tau, (k+1)\tau), \\ \mathbf{u}_{safe}^{k+1,\infty}(t) & \text{for } t \in [(k+1)\tau, \infty), \end{cases} \quad (13)$$

then $\mathbf{y}(t, \mathbf{u}_{desire}^{k,\infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B$ for all t for any $\mathbf{w}(\cdot)$ and $\mathbf{d}(\cdot)$ for any $\mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$.

4.2.2 Correction of the state estimation. Once a new state measurement is received, this function restricts the state estimation so that it is compatible with the state measurement and the state prediction computed at the previous step. Suppose the supervisor returns $\mathbf{u}_{out}^{k-1}(\cdot)$ at time $(k-1)\tau$. Then, at time $k\tau$, we have $\mathbf{x}(k\tau) \in \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))$ by definitions (11) and (12). Also, a new state measurement $\mathbf{x}_m(k\tau)$ is received, which implies $\mathbf{x}(k\tau) \in [\mathbf{x}_m(k\tau) + \delta_{min}, \mathbf{x}_m(k\tau) + \delta_{max}]$. Thus, we make a correction of the state estimation $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$ at time $k\tau$ as the intersection of these two intervals. That is,

$$\begin{aligned} \mathbf{x}^a(k\tau) &= \max(\min \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot)), \mathbf{x}_m(k\tau) + \delta_{min}), \\ \mathbf{x}^b(k\tau) &= \min(\max \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot)), \mathbf{x}_m(k\tau) + \delta_{max}). \end{aligned} \quad (14)$$

Notice that this correction still guarantees $\mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$. The supervisor takes this corrected state estimation as an input as shown in Figure 5.

4.2.3 Safe input generator. Given a feasible schedule, this function generates a corresponding safe input signal. This is possible because the existence of a feasible schedule implies the existence of a safe input signal by Theorem 1. We define a safe input generator $\sigma(\mathbf{X}_{pred}(\mathbf{u}^k(\cdot)), \mathbf{T})$ to compute $\mathbf{u}_{safe}^{k+1,\infty}(\cdot)$, where \mathbf{T} is the schedule returned by $\text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}^k(\cdot)), S)$. For $j \in \mathcal{C}$, if $T_j > 0$,

$$\begin{aligned} \sigma_j(X_{pred,j}(\mathbf{u}_j^k(\cdot)), T_j) &:= u_{safe,j}^{k+1,\infty}(\cdot) \\ &\in \{u_j(\cdot) \in \mathcal{U}_j : \\ &\quad y_j^a(T_j + P_j(T_j), u_j(\cdot), \min X_{pred,j}(\mathbf{u}_j^k(\cdot))) = \beta_j \\ &\quad \text{and } y_j^b(T_j, u_j(\cdot), \max X_{pred,j}(\mathbf{u}_j^k(\cdot))) = \alpha_j\}, \end{aligned} \quad (15)$$

where $\sigma_j(X_{pred,j}(\mathbf{u}_j^k(\cdot)), T_j)$ is the j -th entry of $\sigma(\mathbf{X}_{pred}(\mathbf{u}^k(\cdot)), \mathbf{T})$. If $T_j = 0$, then let $u_{safe,j}^{k+1,\infty}(t) = u_{j,max}$ for $t \in [(k+1)\tau, \infty)$. The safe input signal $u_{safe,j}^{k+1,\infty}(\cdot)$ makes controlled vehicle j enter the intersection no earlier than T_j and exit it no later than $T_j + P_j(T_j)$.

If $\text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), S)$ finds a feasible schedule \mathbf{T} , the supervisor computes a safe input signal $\mathbf{u}_{safe}^{k+1,\infty}(\cdot)$, which is $\mathbf{u}_{safe}^{k+1,\infty}(t)$ restricted to $t \in [(k+1)\tau, (k+2)\tau)$. The supervisor stores this safe input signal for a possible use at the next time step.

4.2.4 Solution of Problem 1 The supervisor is implemented in procedure SUPERVISOR in Algorithm 2.

Algorithm 2 Implementation of the supervisor

```

1: procedure SUPERVISOR( $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k$ )
2:    $(\mathbf{T}_1, ans) = \text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), S)$ 
3:   if  $ans = yes$  and for all  $t \in [0, \tau), B \cap$ 
    $[\mathbf{x}^a(t, \mathbf{u}_{desire}^k(\cdot), \mathbf{x}^a(k\tau)), \mathbf{x}^b(t, \mathbf{u}_{desire}^k(\cdot), \mathbf{x}^b(k\tau))] =$ 
    $\emptyset$  then
4:      $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), \mathbf{T}_1)$ 
5:      $\mathbf{u}_{safe}^{k+1}(\cdot) = \mathbf{u}_{safe}^{k+1, \infty}(t)$  for  $t \in [(k+1)\tau, (k+1)\tau)$ 
6:     return  $\mathbf{u}_{desire}^k(\cdot)$ 
7:   else
8:      $(\mathbf{T}_2, \cdot) = \text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), S)$ 
9:      $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), \mathbf{T}_2)$ 
10:     $\mathbf{u}_{safe}^{k+1}(\cdot) = \mathbf{u}_{safe}^{k+1, \infty}(t)$  for  $t \in [(k+1)\tau, (k+1)\tau)$ 
11:    return  $\mathbf{u}_{safe}^k(\cdot)$ 

```

To initiate the procedure, it is assumed that initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$ and initial desired input \mathbf{u}_{desire}^0 do not cause collisions at any future time. That is, $\text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{desire}^0), S)$ must return *yes* so that $\text{SUPERVISOR}([\mathbf{x}^a(0), \mathbf{x}^b(0)], \mathbf{u}_{desire}^0) \neq \emptyset$.

Theorem 2. Procedure SUPERVISOR in Algorithm 2 implements the supervisor s designed in Problem 1.

Proof. Procedure SUPERVISOR returns $\mathbf{u}_{desire}^k(\cdot)$ in line 6 if ans_1 is *yes*. This implies that there exists $\mathbf{u}_{safe}^{k+1, \infty}(\cdot)$, which in turn, implies by (13) that there exists $\mathbf{u}_{desire}^{k, \infty}(\cdot)$ that satisfies $\mathbf{y}(t, \mathbf{u}_{desire}^{k, \infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B$ for all t for any $\mathbf{w}(\cdot)$ and $\mathbf{d}(\cdot)$ for any $\mathbf{x}(k\tau) \in [\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$. Otherwise, it returns $\mathbf{u}_{safe}^k(\cdot)$ in line 11. This structure corresponds to the supervisor design in Problem 1.

We prove the non-blocking property by mathematical induction on time step k . For the base case, it is assumed that $\text{SUPERVISOR}([\mathbf{x}^a(0), \mathbf{x}^b(0)], \mathbf{u}_{desire}^0) = \mathbf{u}_{out}^0(\cdot) \neq \emptyset$ where $\mathbf{u}_{out}^0(\cdot)$ is defined on time $[0, \tau)$, and $\mathbf{u}_{safe}^{1, \infty}(\cdot)$ is well-defined. We say $\mathbf{u}_{safe}^{1, \infty}(\cdot)$ is well-defined if there exists a schedule \mathbf{T} that defines $\mathbf{u}_{safe}^{1, \infty}(\cdot)$ as $\sigma(\mathbf{X}_{pred}(\mathbf{u}_{desire}^0(\cdot)), \mathbf{T})$. Suppose at $t = (k-1)\tau$, we have $\text{SUPERVISOR}([\mathbf{x}^a((k-1)\tau), \mathbf{x}^b((k-1)\tau)], \mathbf{u}_{desire}^{k-1}(\cdot)) = \mathbf{u}_{out}^{k-1}(\cdot) \neq \emptyset$ and $\mathbf{u}_{safe}^{k, \infty}(\cdot)$ is well-defined. That is, there exists $\mathbf{u}_{safe}^{k, \infty}(\cdot)$ that for all $t \geq 0$ for any $\mathbf{w}(\cdot)$ and $\mathbf{d}(\cdot)$,

$$\forall \mathbf{x}(k\tau) \in \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot)), \quad \mathbf{y}(t, \mathbf{u}_{safe}^{k, \infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B. \quad (16)$$

Then, at $t = k\tau$, we need to show that $\mathbf{u}_{out}^k(\cdot) \neq \emptyset$ no matter what \mathbf{u}_{desire}^k is applied, and $\mathbf{u}_{safe}^{k+1, \infty}(\cdot)$ is well-defined.

In Algorithm 2, SUPERVISOR($[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k$) assigns $\mathbf{u}_{out}^k(\cdot)$ either $\mathbf{u}_{desire}^k(\cdot)$ in line 6 or $\mathbf{u}_{safe}^k(\cdot)$

in line 11. In either case, $\mathbf{u}_{out}^k(\cdot) \neq \emptyset$. In the former case, $ans = yes$ and \mathbf{T}_1 exists, which implies by Theorem 1 that there exists an input signal guaranteeing the avoidance of the Bad set for any uncertainty. Thus, $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), \mathbf{T}_1)$ is well-defined on time $[(k+1)\tau, \infty)$. In the latter case, we consider $\text{EXACT}(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), S)$. Here, $\mathbf{u}_{safe}^k(\cdot)$ is $\mathbf{u}_{safe}^{k, \infty}(\cdot)$ restricted to time $[k\tau, (k+1)\tau)$. If let $\mathbf{u}_{safe}^{k+1, \infty}(\cdot)$ be $\mathbf{u}_{safe}^{k, \infty}(\cdot)$ restricted to time $[(k+1)\tau, \infty)$, we can rewrite (16) as

$$\forall \mathbf{x}((k+1)\tau) \in [\mathbf{x}^a(\tau, \mathbf{u}_{safe}^k(\cdot), \min \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))), \mathbf{x}^b(\tau, \mathbf{u}_{safe}^k(\cdot), \max \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot)))]], \quad \mathbf{y}(t, \mathbf{u}_{safe}^{k+1, \infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}((k+1)\tau)) \notin B. \quad (17)$$

Since by (14), $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)] \subseteq \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))$, the state prediction $\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot))$, which denotes $[\mathbf{x}^a(\tau, \mathbf{u}_{safe}^k(\cdot), \mathbf{x}^a(k\tau)), \mathbf{x}^b(\tau, \mathbf{u}_{safe}^k(\cdot), \mathbf{x}^b(k\tau))]$, satisfies

$$\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)) \subseteq [\mathbf{x}^a(\tau, \mathbf{u}_{safe}^k(\cdot), \min \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot))), \mathbf{x}^b(\tau, \mathbf{u}_{safe}^k(\cdot), \max \mathbf{X}_{pred}(\mathbf{u}_{out}^{k-1}(\cdot)))]], \quad (18)$$

due to the order-preserving property with respect to an initial state in Assumption 1. Thus, (17) is still satisfied for any $\mathbf{x}((k+1)\tau) \in \mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot))$. That is, \mathbf{T}_2 in line 8 exists, and $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), \mathbf{T}_2)$ is well-defined. Therefore, the supervisor is non-blocking. \square

5 Approximate solutions

5.1 Efficient Verification

While scheduling problems on a single machine with arbitrary release times, deadlines, and process times are known to be NP-hard, Garey et al. (1981) proved that the complexity can be reduced to $O(n \log n)$ if process times of all jobs are identical. This was done using “forbidden regions” and the “earliest deadline scheduling (EDD)” rule. Forbidden regions are time intervals during which no feasible job is allowed to start, and can be computed in $O(n \log n)$ time. Once forbidden regions are computed, EDD can be used to solve the scheduling problem in $O(n \log n)$ time.

We design Algorithm 3 by modifying Garey’s result to handle inserted idle-times. We define initial forbidden regions \mathbf{F}_0 to account for the idle-times and set them as inputs of Algorithm 3. In Garey’s result, forbidden regions are initially declared empty and not taken as inputs.

In the procedure in Algorithm 3, r_j, d_j , and t_j denote the j -th entry of \mathbf{r}, \mathbf{d} , and \mathbf{t} , respectively, and $F_{0, \gamma}$ the γ -th interval of \mathbf{F}_0 so that $\mathbf{F}_0 = \cup_{\gamma} F_{0, \gamma}$. Critical time c is the

latest time at which a job can start at each iteration. A set \mathcal{A} contains jobs that have not been scheduled but are ready to be scheduled at time s , which means their release times are smaller than or equal to s . A set \mathcal{B} contains all jobs that have not been yet scheduled. Initially, $\mathcal{B} = \{1, \dots, |\mathbf{r}|\}$ where $|\mathbf{r}|$ is the cardinality of \mathbf{r} .

Algorithm 3 Modified version of the result of Garey et al. (1981)

```

1: procedure POLYNOMIAL( $\mathbf{r}, \mathbf{d}, \mathbf{F}_0$ )
2:   Let  $\sigma$  be a vector of indexes in an increasing order of
    $\mathbf{r}$  and  $\mathbf{F} = \mathbf{F}_0$ .
3:   for  $i = |\mathbf{r}|$  to 1 do  $\triangleright$  Forbidden Region Declaration
4:     for  $j \in \{j : d_j \geq d_{\sigma_i}\}$  do
5:       if  $c_j$  undefined then  $c_j = d_j$ 
6:       else  $c_j = c_j - 1$ 
7:       if  $c_j \in F_\gamma$  for some  $\gamma$  then  $c_j = \inf F_\gamma$ 
8:       if  $\sigma_i = 1$  or  $r_{\sigma_{i-1}} < r_{\sigma_i}$  then
9:          $c = \min\{c_j : c_j \text{ defined}\}$ 
10:      if  $c < r_{\sigma_i}$  then  $ans = no$ 
11:      if  $r_{\sigma_i} \leq c < r_{\sigma_i} + 1$  then  $\mathbf{F} = \mathbf{F} \cup (c - 1, r_{\sigma_i})$ 
12:       $s = 0, \mathcal{A} = \emptyset, \mathcal{B} = \{1, \dots, |\mathbf{r}|\}$   $\triangleright$  Schedule
      Generation
13:      while  $\mathcal{B} \neq \emptyset$  do
14:        if  $s \in F_\gamma$  for some  $\gamma$  then  $s = \sup F_\gamma$ 
15:        if  $\mathcal{A} = \emptyset$  then  $j = \arg \min_{j \in \mathcal{B}} r_j$  and  $t_j = r_j$ 
16:        else  $j = \arg \min_{j \in \mathcal{A}} d_j$  and  $t_j = s$ 
17:         $s = s + 1, \mathcal{B} = \mathcal{B} \setminus \{j\}$  and  $\mathcal{A} = \{k \in \mathcal{B} : r_k \geq s\}$ 
18:       $\pi^*$  = a vector of indexes in an increasing order of  $\mathbf{t}$ 
19:      if  $ans = no$  then
20:        return  $(\emptyset, \pi^*, no)$ 
21:      else return  $(\mathbf{t}, \pi^*, yes)$ 

```

In the following lemma, we prove that procedure POLYNOMIAL in Algorithm 3 solves the IIT scheduling problem with unit process times. This problem is formulated as follows.

Problem 4. Given $\mathbf{r}, \mathbf{d}, \bar{\mathbf{r}}, \bar{\mathbf{p}}$, determine the existence of a schedule \mathbf{t} satisfying

$$\text{for all } j, \quad t_j \in [r_j, d_j], \quad (19)$$

$$\text{for all } i \neq j, \quad (t_i, t_i + 1) \cap (t_j, t_j + 1) = \emptyset, \quad (20)$$

$$\text{for all } j \text{ and } \gamma, \quad (t_j, t_j + 1) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset, \quad (21)$$

where $(\bar{r}_\gamma, \bar{p}_\gamma)$ denotes the γ -th inserted idle-time.

Lemma 1. Procedure POLYNOMIAL in Algorithm 3 solves the IIT scheduling problem with unit process times and finds a feasible schedule if exists.

The proof of Lemma 1 is provided in Appendix B.

To use Algorithm 3, we assign a time interval of equal length to cross the intersection to all vehicles, and formulate a relaxed IIT scheduling problem. The identical process time θ_{max} is defined as follows:

$$\theta_{max} := \max_{j \in \mathcal{C}} \max_{R_j \leq T_j \leq D_j} P_j(T_j). \quad (22)$$

Here, θ_{max} is the maximum time that any controlled vehicle spends crossing an intersection, so that $\theta_{max} \geq P_j(T_j)$ for all $j \in \mathcal{C}$ for any $T_j \in [R_j, D_j]$. In other words, all controlled vehicles are guaranteed to cross the intersection within θ_{max} .

By replacing $P_j(T_j)$ in Problem 3 with θ_{max} , the relaxed IIT scheduling problem is formulated as follows.

Problem 5. (Relaxed IIT scheduling). Given an initial state estimation $[\mathbf{x}^a(0), \mathbf{x}^b(0)]$, determine whether there exists a schedule $\mathbf{T} = (T_1, \dots, T_{n_c}) \in \mathbb{R}_+^{n_c}$ such that for all $j \in \mathcal{C}$,

$$R_j \leq T_j \leq D_j, \quad (23)$$

for all $i \neq j \in \mathcal{C}$ if $T_i, T_j > 0$,

$$(T_i, T_i + \theta_{max}) \cap (T_j, T_j + \theta_{max}) = \emptyset, \quad (24)$$

for all $j \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$ if $T_j > 0$,

$$(T_j, T_j + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset. \quad (25)$$

If $T_j = 0$, $(0, P_j(0))$ replaces $(T_j, T_j + \theta_{max})$ in conditions (24) and (25).

The following algorithm solves this problem by employing POLYNOMIAL. This is an exact solution for Problem 5 by Lemma 1. At line 7, we call $\mathbf{0}$ the zero vector in \mathbb{R}^{n_c} .

Algorithm 4 Exact Solution of Problem 5

```

1: procedure RELAXEDEXACT( $[\mathbf{x}^a(0), \mathbf{x}^b(0)], S$ )
2:   if  $[\mathbf{y}^a(0), \mathbf{y}^b(0)] \cap B \neq \emptyset$  then return  $(\emptyset, \emptyset, no)$ 
3:   for all  $j \in \mathcal{C}$  do calculate  $R_j, D_j, \theta_{max}$ 
4:   for all  $\gamma \in \bar{\mathcal{C}}$  do calculate  $\bar{R}_\gamma, \bar{P}_\gamma$ 
5:    $F_\gamma = (\max(\bar{R}_\gamma/\theta_{max} - 1, 0), \bar{P}_\gamma/\theta_{max})$ 
6:    $\mathcal{M} = \{j \in \mathcal{C} : y_j^b(0) < \alpha_j\}$  and  $\bar{\mathcal{M}} = \mathcal{C} \setminus \mathcal{M}$ 
7:   if  $|\mathcal{M}| = 0$  then return  $(\emptyset, \mathbf{0}, yes)$ 
8:    $T_j = 0$  for all  $j \in \bar{\mathcal{M}}$  and  $P_{max} = \max_{j \in \mathcal{M}} P_j(T_j)$ 
9:   for all  $j \in \mathcal{M}$  do
10:     $r_j = \max(R_j, P_{max})/\theta_{max}$  and  $d_j = D_j/\theta_{max}$ 
11:     $(\mathbf{t}, \pi^*, ans) = \text{POLYNOMIAL}(\mathbf{r}, \mathbf{d}, \mathbf{F})$ 
12:    if  $ans = yes$  then  $T_j = t_j \theta_{max}, \forall j \in \mathcal{M}$ 
13:    return  $(\mathbf{T}, \pi^*, yes)$ 
14:   else return  $(\emptyset, \pi^*, no)$ 

```

In this procedure, all parameters are normalized by θ_{max} because procedure POLYNOMIAL assumes unit process times. In line 5, the idle-time $(\bar{R}_\gamma, \bar{P}_\gamma)$ is translated into an initial set of forbidden regions F_γ so that condition (25) is equivalent to $t_j \notin F_\gamma$. If $t_j \in F_\gamma$, then $T_j \in (\max(\bar{R}_\gamma - \theta_{max}, 0), \bar{P}_\gamma)$ so that either $T_j \in (\bar{R}_\gamma, \bar{P}_\gamma)$ or $T_j + \theta_{max} \in (\bar{R}_\gamma, \bar{P}_\gamma)$.

The running time of procedure RELAXEDEXACT is dominated by POLYNOMIAL in line 11, which has an asymptotic running time of $O(n_c^2)$.

By exploiting procedure RELAXEDEXACT in Algorithm 4, we design a new procedure, called APPROX, that solves Problem 2 within an approximation bound. This procedure schedules vehicles according to a sequence returned by procedure RELAXEDEXACT, thereby inheriting computational efficiency. This sequence is denoted by π^* in the following algorithm.

Algorithm 5 Approximate Solution of Problem 3

- 1: **procedure** APPROX($[\mathbf{x}^a(0), \mathbf{x}^b(0)], S$)
 - 2: **if** $[\mathbf{y}^a(0), \mathbf{y}^b(0)] \cap B \neq \emptyset$ **then return** (\emptyset, no)
 - 3: **if** $y_j^b(0) \geq \alpha_j$ for all $j \in \mathcal{C}$ **then return** (\emptyset, yes)
 - 4: $(\cdot, \pi^*, \cdot) = \text{RELAXEDEXACT}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S)$
 - 5: $(\mathbf{T}, ans) = \text{SCHEDULING}(\pi^*, [\mathbf{x}^a(0), \mathbf{x}^b(0)], S)$
-

Procedure APPROX in Algorithm 5 trades exactness for computational efficiency. We will prove that procedure APPROX is more conservative than procedure EXACT, that is, there exists an instance $I = ([\mathbf{x}^a(0), \mathbf{x}^b(0)], S)$ such that APPROX(I) returns *no* while EXACT(I) returns *yes*. In order to quantify the degree of conservatism, we prove two theorems. The first theorem states that if procedure APPROX returns *yes*, then there exists an input signal to avoid the Bad set. In the second theorem, if procedure APPROX returns *no*, then there does not exist an input signal to avoid an inflated Bad set, which accounts for the conservatism.

Theorem 3. *If* $\text{APPROX}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\mathbf{T}, yes)$, *then* $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) \in \text{Problem 2}$.

Proof. By Theorem 1, we only need to show that $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) \in \text{Problem 3}$. Notice that procedure APPROX of Algorithm 5 returns *yes* in the case of line 3. In this case, $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) \in \text{Problem 3}$ since all vehicles have already crossed the intersection. The procedure also returns *yes* when procedure SCHEDULING given a sequence π^* returns *yes* in line 5. Since $\pi^* \in \mathcal{P}$, where \mathcal{P} is a set of all permutations, and π^* yields a feasible schedule, we know EXACT($[\mathbf{x}^a(0), \mathbf{x}^b(0)], S$) returns *yes*. Thus, $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) \in \text{Problem 3}$.

However, the converse of Theorem 3 is not true. That is, for some instances such that $([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) \in$

Problem 2, procedure APPROX can return *no*. To consider these instances, we introduce an *inflated* Bad set.

The IIT scheduling problem finds a schedule satisfying $y_j^b(T_j) = \alpha_j$ and $y_j^a(T_j + P_j(T_j)) = \beta_j$ for $j \in \mathcal{C}$. In contrast, the relaxed IIT scheduling problem finds a schedule satisfying $y_j^b(T_j) = \alpha_j$ and $y_j^a(T_j + \theta_{max}) \geq \beta_j$. Given that the farthest distance that controlled vehicle j can travel during θ_{max} is $\theta_{max}v_{j,max}$, we define an “inflated” intersection $(\alpha_j, \hat{\beta}_j)$ such that $\hat{\beta}_j := \alpha_j + \theta_{max}v_{j,max}$. Notice that $\beta_j \leq \hat{\beta}_j$. Because the process times are only defined for controlled vehicles, $\hat{\beta}_\gamma = \beta_\gamma$ for $\gamma \in \bar{\mathcal{C}}$. Thus, inflated Bad set \hat{B} is defined as follows:

$$\hat{B} := \{\mathbf{y} \in Y : y_i \in (\alpha_i, \hat{\beta}_i) \text{ and } y_j \in (\alpha_j, \hat{\beta}_j) \text{ for some } i \neq j \text{ such that } i \in \mathcal{C} \cup \bar{\mathcal{C}} \text{ and } j \in \mathcal{C}\}. \quad (26)$$

By replacing Bad set B in Problem 2 with inflated Bad set \hat{B} , we can formulate the relaxed verification problem. The following theorem is a key result that shows the approximation bound of procedure APPROX.

Lemma 2. *If* $\text{APPROX}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\mathbf{T}, yes)$, *and* $\text{RELAXEDEXACT}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\bar{\mathbf{T}}, \pi^*, yes)$, *then* $T_j \leq \bar{T}_j$ for all $j \in \mathcal{C}$.

Lemma 3. *If* $\text{APPROX}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\emptyset, no)$, *then* $\text{RELAXEDEXACT}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\emptyset, \pi^*, no)$.

The proofs of Lemmas 2 and 3 can be found in Appendix B.

Theorem 4. *If* $\text{APPROX}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\emptyset, no)$, *then there is no input signal* $\mathbf{u}(\cdot)$ *that guarantees* $\mathbf{y}(t, \mathbf{u}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(0)) \notin \hat{B}$ *for all* $t \geq 0$ *for any* $\mathbf{w}(\cdot) \in \mathcal{W}$, $\mathbf{d}(\cdot) \in \mathcal{D}$, *and* $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$.

Proof. By Lemma 3, $\text{APPROX}([\mathbf{x}^a(0), \mathbf{x}^b(0)], S) = (\emptyset, no)$ means that RELAXEDEXACT returns *no*. Thus, we will prove that if there is no schedule satisfying conditions (23)-(25), there is no input signal to avoid the inflated Bad set for any uncertainty. For ease of proof, we prove the contrapositive statement. That is, assume that there is an input signal $\tilde{\mathbf{u}}(\cdot) \in \mathcal{U}$ such that $\mathbf{y}(t, \tilde{\mathbf{u}}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(0)) \notin \hat{B}$ for all t for any $\mathbf{w}(\cdot) \in \mathcal{W}$, $\mathbf{d}(\cdot) \in \mathcal{D}$, and $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$. Then, there exists schedule $\tilde{\mathbf{T}}$ satisfying conditions (23)-(25). This proof is similar to the first part of the proof of Theorem 1.

For all $j \in \mathcal{C}$, define \tilde{T}_j as $y_j^b(\tilde{T}_j, \tilde{u}_j(\cdot), x_j^b(0)) = \alpha_j$ if $y_j^b(0) < \alpha_j$, and 0 otherwise, and \tilde{K}_j as $y_j^a(\tilde{K}_j, \tilde{u}_j(\cdot), x_j^a(0)) = \hat{\beta}_j$ if $y_j^a(0) < \hat{\beta}_j$, and 0 otherwise. For $i \neq j \in \mathcal{C}$, since $y_i(t, \tilde{u}_i(\cdot), d_i(\cdot), x_i(0))$ and $y_j(t, \tilde{u}_j(\cdot), d_j(\cdot), x_j(0))$ avoid the inflated Bad set for any $\mathbf{d}(\cdot) \in \mathcal{D}$ and $\mathbf{x}(0) \in [\mathbf{x}^a(0), \mathbf{x}^b(0)]$, $(\tilde{T}_i, \tilde{K}_i) \cap (\tilde{T}_j, \tilde{K}_j) = \emptyset$. Since the inflated intersection

Algorithm 6 Implementation of s_e

```

1: procedure EFFSUPERVISOR( $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k$ )
2:    $(\mathbf{T}_1, ans_1) = \text{APPROX}(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), S)$ 
3:   if  $ans_1 = \text{yes}$  and for all  $t \in [0, \tau)$ ,  $B \cap [\mathbf{x}^a(t, \mathbf{u}_{desire}^k(\cdot), \mathbf{x}^a(k\tau)), \mathbf{x}^b(t, \mathbf{u}_{desire}^k(\cdot), \mathbf{x}^b(k\tau))] = \emptyset$  then
4:      $\pi^k =$  a vector of indexes in the increasing order of nonzero entries of  $\mathbf{T}_1$ 
5:      $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{desire}^k(\cdot)), \mathbf{T}_1)$ 
6:      $\mathbf{u}_{safe}^{k+1}(\cdot) = \mathbf{u}_{safe}^{k+1, \infty}(t)$  for  $t \in [(k+1)\tau, (k+2)\tau)$ 
7:     return  $\mathbf{u}_{desire}^k(\cdot)$ 
8:   else
9:      $(\mathbf{T}_2, ans_2) = \text{APPROX}(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), S)$ 
10:    if  $ans_2 = \text{no}$  then
11:       $(\mathbf{T}_2, ans_3) = \text{SCHEDULING}(\pi^{k-1}, \mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), S)$ 
12:       $\pi^k =$  a vector of indexes in the increasing order of nonzero entries of  $\mathbf{T}_2$ 
13:       $\mathbf{u}_{safe}^{k+1, \infty}(\cdot) = \sigma(\mathbf{X}_{pred}(\mathbf{u}_{safe}^k(\cdot)), \mathbf{T}_2)$ 
14:       $\mathbf{u}_{safe}^{k+1}(\cdot) = \mathbf{u}_{safe}^{k+1, \infty}(t)$  for  $t \in [(k+1)\tau, (k+2)\tau)$ 
15:      return  $\mathbf{u}_{safe}^k(\cdot)$ 

```

takes account of the maximum driving distance during θ_{max} , we have $y_j^a(\tilde{T}_j + \theta_{max}, \tilde{u}_j(\cdot), x^a(0)) \leq \hat{\beta}_j$. This implies $\tilde{T}_j + \theta_{max} \leq \tilde{K}_j$ for all $j \in \mathcal{C}$ because $y_j^a(t)$ is non-decreasing in t . Thus, $(\tilde{T}_i, \tilde{T}_i + \theta_{max}) \cap (\tilde{T}_j, \tilde{T}_j + \theta_{max}) = \emptyset$ (condition (24)).

For $\gamma \in \bar{\mathcal{C}}$, define $(\bar{R}_\gamma, \bar{P}_\gamma)$ as in Definition 2. Then, for any $w_\gamma(\cdot) \in \mathcal{W}_\gamma$, $d_\gamma(\cdot) \in \mathcal{D}_\gamma$, and $x_\gamma(0) \in [x_\gamma^a(0), x_\gamma^b(0)]$, uncontrolled vehicle γ enters the intersection no earlier than \bar{R}_γ and exits it no later than \bar{P}_γ . In order for $\tilde{u}_j(\cdot)$ to guarantee that $y_j(t, \tilde{u}_j(\cdot), d_j(\cdot), x_j(0))$ and $y_\gamma(t)$ never meet inside the intersection for any uncertainty, $(\tilde{T}_j, \tilde{K}_j) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. Since $\tilde{T}_j + \theta_{max} \leq \tilde{K}_j$, we have $(\tilde{T}_j, \tilde{T}_j + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$ (condition (25)).

Condition (23) is satisfied by the definitions of R_j and D_j . \square

In summary, Theorem 3 states that if procedure APPROX returns *yes*, there exists an input signal to avoid the Bad set B for any uncertainty. As stated in Theorem 4, if the procedure returns *no*, there does not exist an input signal to avoid the inflated Bad set \hat{B} for any uncertainty. Thus, the difference between B and \hat{B} is a measure of the approximation bound of procedure APPROX.

5.2 Efficient supervisor

In order for a supervisor to run in real time, the verification problem must be solved within the time step τ . However, if a large number of controlled vehicles is considered, the problem becomes intractable. Thus, we employ the results in Section 5.1 to design efficient supervisors. The simplest solution would be to implement a supervisor

$\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{desire}^k)$ defined as follows:

$$\hat{s} = \begin{cases} \mathbf{u}_{desire}^k(\cdot) & \text{if } \exists \mathbf{u}_{desire}^{k, \infty}(\cdot) : \text{for all } t \geq 0 \\ & \mathbf{y}(t, \mathbf{u}_{desire}^{k, \infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin \hat{B} \\ \mathbf{u}_{safe}^k(\cdot) & \text{otherwise.} \end{cases}$$

The only difference from the exact supervisor s is that the Bad set B is replaced by the inflated Bad set \hat{B} .

Instead of implementing \hat{s} , we consider a procedure implementing another supervisor, s_e , by using procedure APPROX to verify whether or not to override drivers. We call this procedure EFFSUPERVISOR in Algorithm 6.

Notice that this procedure stores the feasible sequence of vehicles crossing the intersection at every time step (lines 4 and 12). This is because when the sequence considered in line 9 does not yield a feasible schedule, the previous step's sequence π^{k-1} can be used to generate one. This is where in procedure SCHEDULING in Algorithm 1, $|\pi_0|$ can be different from $|\pi|$, where $\pi_0 = \pi^{k-1}$ and $\pi \in \mathbb{R}^{|\mathcal{M}|}$. This is because \mathcal{M} is a set of vehicles that have not entered an intersection at the current step, whereas π^{k-1} is a sequence of vehicles that had not entered an intersection at the previous step.

The fact that the previous step's sequence leads to a feasible schedule ensures the non-blocking property of the supervisor and is proved in the next theorem. Bruni et al. (2013) proposed an efficient supervisor considering measurement errors and unmodeled dynamics with all vehicles controlled. Their supervisor cannot find a feasible schedule at every step and thus uses the previous schedule until a new feasible schedule is found. This ignores the correction of the state estimation during the open-loop control, thereby being more conservative than our efficient

supervisor, which updates the schedule based on the most current state estimation.

Procedure EFFSUPERVISOR takes polynomial computation time, and guarantees avoiding the Bad set B because such a safe input exists if procedure APPROX returns *yes* by Theorem 3. Most importantly, we will prove in the following theorem that s_e is less restrictive than \hat{s} , in the sense that \hat{s} overrides controlled vehicles more frequently than s_e .

Theorem 5. *The supervisor s_e is less restrictive than \hat{s} : if $\text{EFFSUPERVISOR}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{\text{desire}}^k) = \mathbf{u}_{\text{safe}}^k(\cdot)$, then $\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{\text{desire}}^k) = \mathbf{u}_{\text{safe}}^k(\cdot)$. Moreover, s_e is non-blocking.*

Proof. $\text{EFFSUPERVISOR}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{u}_{\text{desire}}^k) = \mathbf{u}_{\text{safe}}^k(\cdot)$ when $\text{APPROX}(\mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{desire}}^k(\cdot)), S)$ returns *no*. By Theorem 4, there is no input signal $\mathbf{u}_{\text{desire}}^k(\cdot) \in \mathcal{U}$ that can prevent entering the inflated Bad set for some uncertainties. Thus, $\hat{s}(\mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{desire}}^k), S)$ returns $\mathbf{u}_{\text{safe}}^k(\cdot)$. This proves that s_e is less restrictive than \hat{s} .

The non-blocking property is proved by mathematical induction on time step k . For the base case, it is assumed that $\text{EFFSUPERVISOR}([\mathbf{x}^a(0), \mathbf{x}^b(0)], \mathbf{u}_{\text{desire}}^0) = \mathbf{u}_{\text{out}}^0(\cdot) \neq \emptyset$ where $\mathbf{u}_{\text{out}}^0(\cdot)$ is defined on time $[0, \tau)$, and $\mathbf{u}_{\text{safe}}^{1,\infty}(\cdot)$ is well-defined, that is, there exists a schedule \mathbf{T} that defines $\mathbf{u}_{\text{safe}}^{1,\infty}(\cdot)$ as $\sigma(\mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{desire}}^0(\cdot)), \mathbf{T})$. Suppose at $t = (k-1)\tau$, we have $\text{EFFSUPERVISOR}([\mathbf{x}^a((k-1)\tau), \mathbf{x}^b((k-1)\tau)], \mathbf{u}_{\text{desire}}^{k-1}) = \mathbf{u}_{\text{out}}^{k-1}(\cdot) \neq \emptyset$ and $\mathbf{u}_{\text{safe}}^{k,\infty}(\cdot)$ is well-defined, which satisfies $\mathbf{y}(t, \mathbf{u}_{\text{safe}}^{k,\infty}(\cdot), \mathbf{w}(\cdot), \mathbf{d}(\cdot), \mathbf{x}(k\tau)) \notin B$ for any $\mathbf{w}(\cdot) \in \mathcal{W}, \mathbf{d}(\cdot) \in \mathcal{D}$, and $\mathbf{x}(k\tau) \in \mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{out}}^{k-1}(\cdot))$. Now, we need to prove that $\mathbf{u}_{\text{out}}^k(\cdot) \neq \emptyset$, and $\mathbf{u}_{\text{safe}}^{k+1,\infty}(\cdot)$ is well-defined.

In Algorithm 6, if $\text{ans}_1 = \text{yes}$ in line 2 or $\text{ans}_2 = \text{yes}$ in line 9, by Theorem 3, there exists an input signal that makes the vehicle trajectories avoid entering the Bad set. Thus, $\mathbf{u}_{\text{safe}}^{k+1,\infty}(\cdot)$ is well-defined on time $[(k+1)\tau, \infty)$ in lines 5 and 13. Also, $\mathbf{u}_{\text{out}}^k(\cdot) = \emptyset$ because $\mathbf{u}_{\text{out}}^k(\cdot) = \mathbf{u}_{\text{desire}}^{k,\infty}(\cdot)$ or $\mathbf{u}_{\text{out}}^k(\cdot) = \mathbf{u}_{\text{safe}}^k(\cdot)$.

Suppose that $\text{ans}_1 = \text{no}$ and $\text{ans}_2 = \text{no}$. Then, given π^{k-1} , we need to prove that $\text{ans}_3 = \text{yes}$ and \mathbf{T}_2 exists in line 11. Notice that π^{k-1} is a vector of indexed in the increasing order of the nonzero entries of a feasible schedule \mathbf{T}^{k-1} of the previous step. That is, at the previous step, $\text{SCHEDULING}(\pi^{k-1}, \mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{out}}^{k-1}(\cdot)), S) = (\mathbf{T}^{k-1}, \text{yes})$. We will show that the existence of \mathbf{T}^{k-1} implies that of \mathbf{T}_2 .

Let $\mathbf{u}_{\text{safe}}^{k+1,\infty}(\cdot)$ be $\mathbf{u}_{\text{safe}}^{k,\infty}(\cdot)$ restricted to time $[(k+1)\tau, \infty)$ and $\mathbf{u}_{\text{safe}}^k(\cdot)$ be $\mathbf{u}_{\text{safe}}^{k,\infty}(\cdot)$ restricted to time $[k\tau, (k+1)\tau)$. Then, the j -th entry of \mathbf{T}^{k-1} is as follows:

1) τ). Then, the j -th entry of \mathbf{T}^{k-1} is as follows:

$$\begin{aligned} T_j^{k-1} &:= \{t : y_j^b(t, u_{\text{safe},j}^{k,\infty}(\cdot), \max X_{\text{pred},j}(u_{\text{out},j}^{k-1}(\cdot))) = \alpha_j\} \\ &= \{t : y_j^b(t, u_{\text{safe},j}^{k+1,\infty}(\cdot), \\ &\quad x^b(\tau, u_{\text{safe},j}^k(\cdot), \max X_{\text{pred},j}(u_{\text{out},j}^{k-1}(\cdot))) = \alpha_j\} + \tau \\ &\leq \{t : y_j^b(t, u_{\text{safe},j}^{k+1,\infty}(\cdot), x^b(\tau, u_{\text{safe},j}^k(\cdot), x^b(k\tau))) = \alpha_j\} + \tau \\ &:= \tilde{T}_j + \tau. \end{aligned} \quad (27)$$

The inequality is due to $[x_j^a(k\tau), x_j^b(k\tau)] \subseteq X_{\text{pred},j}(u_{\text{out},j}^{k-1}(\cdot))$ by (14) and the order-preserving property in Assumption 1. Similarly, its process time $T_j^{k-1} + P_j^{k-1}(T_j^{k-1})$ is as follows:

$$\begin{aligned} T_j^{k-1} + P_j^{k-1}(T_j^{k-1}) &:= \{t : y_j^a(t, u_{\text{safe},j}^{k,\infty}(\cdot), \min X_{\text{pred},j}(u_{\text{out},j}^{k-1}(\cdot))) = \beta_j\} \\ &= \{t : y_j^a(t, u_{\text{safe},j}^{k+1,\infty}(\cdot), \\ &\quad x^a(\tau, u_{\text{safe},j}^k(\cdot), \min X_{\text{pred},j}(u_{\text{out},j}^{k-1}(\cdot))) = \beta_j\} + \tau \\ &\geq \{t : y_j^a(t, u_{\text{safe},j}^{k+1,\infty}(\cdot), x^a(\tau, u_{\text{safe},j}^k(\cdot), x^a(k\tau))) = \beta_j\} + \tau \\ &:= \tilde{T}_j + \tilde{P}_j(\tilde{T}_j) + \tau. \end{aligned} \quad (28)$$

For $\gamma \in \bar{\mathcal{C}}$, let $(\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1})$ denote the idle-time given $\mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{out}}^{k-1}(\cdot))$.

We now show that the schedule $\tilde{\mathbf{T}} := (\tilde{T}_j : j \in \mathcal{C})$ is a feasible schedule of $\text{SCHEDULING}(\pi^{k-1}, \mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{safe}}^k(\cdot)), S)$. The release time and deadline of this procedure for $j \in \mathcal{C}$ is by definition,

$$\begin{aligned} R_j &= \min_{u_j(\cdot) \in \mathcal{U}_j} \{t : y_j^b(t, u_j(\cdot), \max X_{\text{pred},j}(u_{\text{safe},j}^k(\cdot))) = \alpha_j\}, \\ D_j &= \max_{u_j(\cdot) \in \mathcal{U}_j} \{t : y_j^b(t, u_j(\cdot), \max X_{\text{pred},j}(u_{\text{safe},j}^k(\cdot))) = \alpha_j\}. \end{aligned}$$

Since $\mathbf{x}^b(\tau, \mathbf{u}_{\text{safe}}^k(\cdot), \mathbf{x}^b(k\tau)) = \max \mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{safe}}^k(\cdot))$ in (11), we have $\tilde{T}_j \in [R_j, D_j]$.

Since \mathbf{T}^{k-1} is feasible, $(T_j^{k-1}, T_j^{k-1} + P_j^{k-1}(T_j^{k-1})) \cap (T_i^{k-1}, T_i^{k-1} + P_i^{k-1}(T_i^{k-1})) = \emptyset$ for $i \neq j \in \mathcal{C}$. Because of (27) and (28),

$$(\tilde{T}_j, \tilde{T}_j + \tilde{P}_j(\tilde{T}_j)) \subseteq (T_j^{k-1}, T_j^{k-1} + P_j^{k-1}(T_j^{k-1})) - \tau,$$

for all $j \in \mathcal{C}$. Thus, $(\tilde{T}_j, \tilde{T}_j + \tilde{P}_j(\tilde{T}_j)) \cap (\tilde{T}_i, \tilde{T}_i + \tilde{P}_i(\tilde{T}_i)) = \emptyset$.

Similarly for $\gamma \in \bar{\mathcal{C}}$, it is not difficult to see that the idle-time of $\text{SCHEDULING}(\pi^{k-1}, \mathbf{X}_{\text{pred}}(\mathbf{u}_{\text{safe}}^k(\cdot)), S)$ denoted by $(\bar{R}_\gamma, \bar{P}_\gamma)$ becomes a subset of $(\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1}) - \tau$. Since $(T_j^{k-1}, T_j^{k-1} + P_j^{k-1}(T_j^{k-1})) \cap (\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1}) = \emptyset$, we have $(\tilde{T}_j, \tilde{T}_j + \tilde{P}_j(\tilde{T}_j)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$.

Thus, $\tilde{\mathbf{T}}$ is feasible, thereby implying $\text{ans}_3 = \text{yes}$ in line 11 of Algorithm 6. Since $\mathbf{T}_2 = \tilde{\mathbf{T}}$ exists, $\mathbf{u}_{\text{safe}}^{k+1,\infty}(\cdot)$ is well-defined.

Therefore, the supervisor is non-blocking. \square

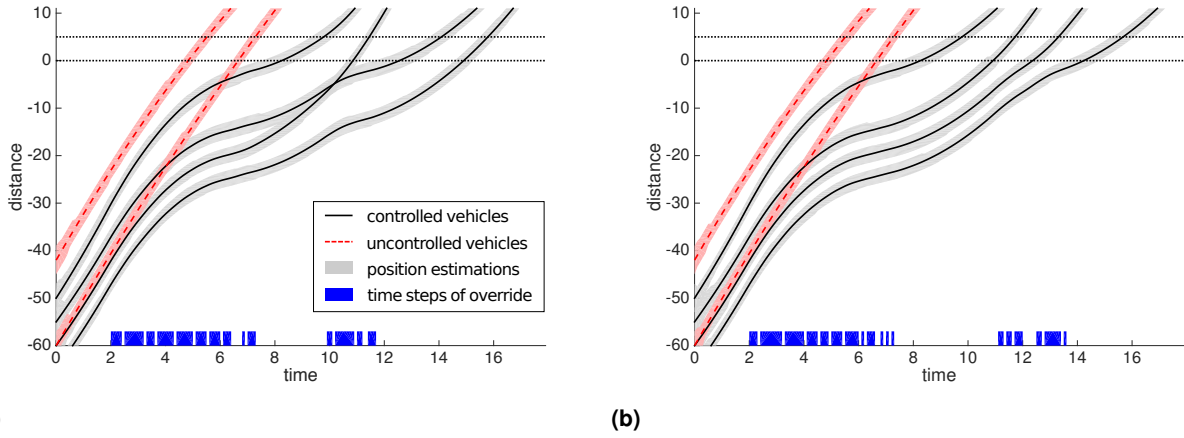


Figure 6. Simulation of the exact supervisor s in (a) and the efficient supervisor s_e in (b). The dotted lines at distance 0 and 5 represent the location of the intersection. The solid black lines and the dotted red lines are the exact trajectories of the controlled and uncontrolled vehicles, respectively, and the shading around the lines represent the position estimation. The blue boxes on the bottom are the time steps at which the supervisor overrides the controlled vehicles. The initial exact state is $\mathbf{y}(0) = (-42, -50, -55, -60, -60, -65)$ and $\mathbf{v}(0) = (10, 9, 8, 8, 10, 8)$.

In summary, the efficient supervisors \hat{s} and s_e are more restrictive than the exact supervisor s , and \hat{s} is more restrictive than s_e by Theorem 5.

6 Simulations

In this section, we present simulation results of the exact and efficient supervisors, s and s_e , in two scenarios by running Algorithms 2 and 6. These are implemented using MATLAB on a personal computer with an 3.10GHz Intel Core i7-3770s processor with 8 GB RAM.

With the dynamic states $x_j = (y_j, v_j)$ and $x_\gamma = (y_\gamma, v_\gamma)$ for $j \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$, the vehicle dynamics considered in the simulation are as follows:

$$\begin{aligned} \dot{y}_j &= v_j + d_{y,j}, \\ \dot{v}_j &= \begin{cases} \max(0, u_j - bv_j^2 + d_{v,j}) & \text{if } (v_j = v_{j,min}), \\ \min(0, u_j - bv_j^2 + d_{v,j}) & \text{if } (v_j = v_{j,max}), \\ u_j - bv_j^2 + d_{v,j} & \text{otherwise,} \end{cases} \end{aligned}$$

and

$$\begin{aligned} \dot{y}_\gamma &= v_\gamma + d_{y,\gamma}, \\ \dot{v}_\gamma &= \begin{cases} \max(0, w_\gamma - bv_\gamma^2 + d_{v,\gamma}) & \text{if } (v_\gamma = v_{\gamma,min}), \\ \min(0, w_\gamma - bv_\gamma^2 + d_{v,\gamma}) & \text{if } (v_\gamma = v_{\gamma,max}), \\ w_\gamma - bv_\gamma^2 + d_{v,\gamma} & \text{otherwise,} \end{cases} \end{aligned}$$

where $b = 0.001$ is a drag coefficient, and $d_{y,j}, d_{y,\gamma}, d_{v,i}, d_{v,\gamma}$ are disturbances representing unmodeled dynamics bounded by -0.05 and 0.05 . The speeds v_j, v_γ are bounded by $v_{j,min} = v_{\gamma,min} = 1.39$ and $v_{j,max} = v_{\gamma,max} = 13.9$, the input u_j by $u_{j,min} = -2.5$

and $u_{j,max} = 2.5$, and the driver-input w_γ by $w_{\gamma,min} = -0.5$ and $w_{\gamma,max} = 0.5$. In the simulation, the disturbances and the driver-inputs of uncontrolled vehicles are randomly chosen within their bounds.

At each time step $\tau = 0.1$, the supervisors determine whether there are impending collisions at an intersection located at $(\alpha_i, \beta_i) = (0, 5)$ for all $i \in \mathcal{C} \cup \bar{\mathcal{C}}$. The states of all vehicles are measured subject to noises, $\delta_{y,i} \in [-3, 3]$ and $\delta_{v,i} \in [-0.05, 0.05]$. For the sake of simplicity, in the simulation, we let $u_{j,dessire}^k = 1$ for all vehicles at all times.

In the first scenario, four controlled and two uncontrolled vehicles are approaching an intersection as in Figure 6, which illustrates the position trajectories of the vehicles in time.

The simulation result of the exact supervisor s is shown in Figure 6a, and that of the efficient supervisor s_e is shown in Figure 6b. Though we said in Section 5.2 that s_e is more conservative than s – in the sense that given the same initial condition, s evaluates more inputs to find one guaranteeing avoiding the Bad set than s_e – comparison of the override time steps (blue boxes) in Figures 6a and 6b indicates that s_e does not override the drivers more than s . It takes 0.067 s and 0.011 s per iteration to execute s (Algorithm 2) and s_e (Algorithm 6), respectively, in the worst case. This validates that the computation of the efficient supervisor is faster than that of the exact supervisor. In both simulations, the intersection at distance $(0, 5)$ is occupied by one vehicle at a time.

The second scenario considers twelve controlled and two uncontrolled vehicles. Due to the large number of controlled vehicles involved, the exact supervisor cannot solve the verification problem within one time step. Thus, the only

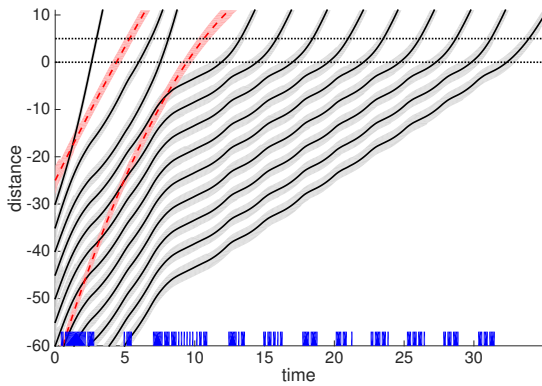


Figure 7. Simulation of the efficient supervisor s_e . The initial exact state is $y(0) = (-25, -30, -35, -40, -45, -50, -55, -60, -65, -65, -70, -75, -80, -85)$ and $v(0) = (6, 9, 9, 8, 8, 8, 8, 8, 8, 9.5, 8, 8, 8, 8)$.

option to resolve conflict in this scenario is to implement the efficient supervisor (Algorithm 6). As shown in Figure 7, the efficient supervisor assists the vehicles to prevent any collision at the intersection. In the worst case, it takes 0.033 s per iteration.

7 Experiments

In this section, we describe the experimental validation of the exact supervisor on an intersection testbed. First, we introduce the laboratory apparatus. Then, we describe the dynamic model of the RC cars used in the experiment and the techniques adopted to reduce disturbances. The results of the experiment are provided at the end of this section.

7.1 Experimental setup

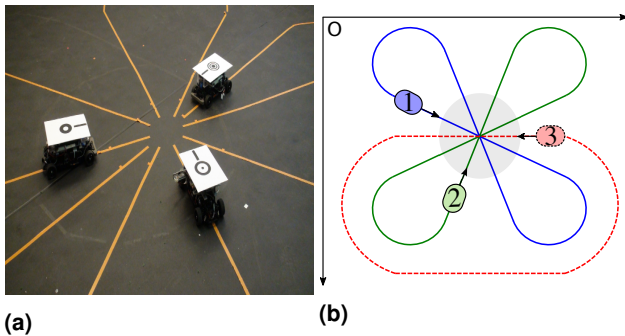


Figure 8. (a) The RC cars used in the experiment. The camera system detects their symbols to identify each car and measure its position. (b) The paths of the cars on the testbed. Cars 1 and 2 are controlled by the supervisor while car 3 is not. A car is considered to be occupying the intersection when its position is within the shaded circle.

The cars used for this experiment are each built onto a Tamiya scaled RC car chassis equipped with a DC motor. A micro-controller (Acroname Moto 1.0) is used to control the steering servo and the motor through two separate PWM channels. An on-board computer (Mini-ITX running Fedora) runs the C programs, which provide the functionalities required to communicate with the centralized supervisor and to control the micro-controller. The system is powered by two batteries (Tenergy Li-Ion 14.8 V 4400 mAh) connected to a capacitor (Aluminium Electrolytic Capacitor 12000 uF 25 V) through a power relay (Omron G5SB). A power amplifier (Acroname 3 A Back EMF H-bridge) connected to the batteries through a switch provides necessary power to the motor.

During the experiment, three cars follow distinct paths intersecting at a single point on a 6 m \times 6 m testbed as shown in Figure 8b. Each path is stored as sequential points on the coordinate system illustrated in the figure. Cars 1 and 2 are controlled by the exact supervisor when necessary, while car 3 is not controllable. We program cars 1 and 2 to maintain constant motor input, which corresponds to the desired input, while car 3 is driven by a human operator.

Each car has access to its own wheel speed through a quadrature encoder mounted on the rear axle. The position and direction of the cars are measured by an over-head vision system, which comprises six cameras on the ceiling and three computers processing images taken by the cameras. Each car is identified by a symbol attached to its roof as shown in Figure 8a. The measurement of the speeds, positions, and directions of all cars are collected by an external computer via a wireless connection of the 802.11b standard using UDP/IP. Then, the computer distributes these measurements to the controlled cars, together with other information such as the desired inputs of the controlled vehicles and model parameters discussed in the next section.

7.2 Car dynamics model

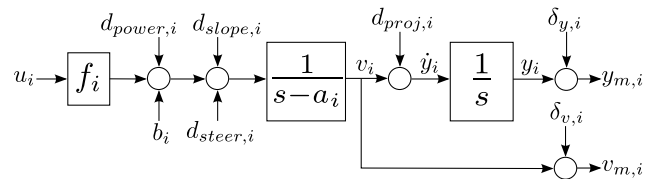


Figure 9. Block diagram of the RC car model. The model includes the measurement disturbances $\delta_{y,i}$ and $\delta_{v,i}$ on the car position and speed, respectively.

A block diagram in Figure 9 represents the car dynamics. In this section, we design a compensating input to reduce the effects of the disturbances $d_{proj,i}$, $d_{power,i}$, $d_{slope,i}$, and $d_{steer,i}$.

Before compensating for the disturbances, the dynamics of the RC cars are as follows: for car $i \in \mathcal{C} \cup \bar{\mathcal{C}}$ where $\mathcal{C} = \{1, 2\}$ and $\bar{\mathcal{C}} = \{3\}$,

$$\begin{aligned}\dot{y}_i &= v_i + d_{proj,i}, \\ \dot{v}_i &= a_i v_i + b_i + f_i u_i + d_{power,i} + d_{slope,i} + d_{steer,i}\end{aligned}\quad (29)$$

where y_i is the longitudinal position of car i along the path, v_i is its wheel speed, and u_i is the motor input with model parameters a_i, b_i , and f_i . The disturbances $d_{proj,i}, d_{power,i}, d_{slope,i}$, and $d_{steer,i}$ are explained below and compensated to reduce their effects.

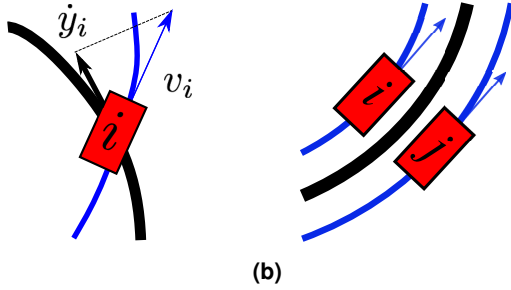


Figure 10. Cases when $d_{proj,i} \neq 0$. Imperfect path following contributes to the discrepancy between the longitudinal speed \dot{y}_i and the wheel speed v_i .

Notice that v_i is the car's wheel speed while \dot{y}_i is the projected speed on the longitudinal path of car i . It is possible that $\dot{y}_i \neq v_i$ when the car does not follow its path exactly, and this discrepancy is represented by the term $d_{proj,i}$. Figure 10 shows two cases in which $\dot{y}_i \neq v_i$. In Figure 10a, the direction of car i differs from the tangent of the longitudinal path (black line) so that $v_i \neq \dot{y}_i$. In Figure 10b, two cars follow the same path with the same wheel speed v but with slight deviation from the path, causing $\dot{y}_i > v > \dot{y}_j$.

This dynamic model (29) includes three different disturbances on the acceleration. The disturbance $d_{power,i}$ describes the first-order dynamic behavior empirically observed in the motor as a consequence of the power connection. By running the cars in a circle with constant motor input for several minutes, we can model $d_{power,i} = g_i e^{-t/h_i} u'_i$, where g_i is the gain and h_i the time constant. The car-specific parameters g_i and h_i are estimated by analyzing the collected data using the least square method (see Rizzi (2014) for the data).

The disturbance $d_{slope,i}$ is introduced to model the slope of the testbed, which is not completely flat and has non-negligible effects on the car speed. The disturbance $d_{steer,i}$ takes into account the fact that the steering and motor dynamics are coupled (Verma et al. (2008)). Since the testbed slope and the steering input are approximately the same at the same point of the path, we estimate these two

disturbances as a path-dependent function $d_{path,i}(y) := d_{steer,i}(y) + d_{slope,i}(y)$. This function is different for each path and estimated by running the car multiple times and curve fitting of the obtained data.

Eliminating the effects of these disturbances is critical because otherwise it is difficult to initiate Algorithm 2 with a feasible initial condition, especially in a spatially constrained environment, such as a laboratory testbed. To this end, we introduce a compensating term $c_i(t, y_i)$ to the motor input so that $u_i = u'_i + c_i(t, y_i)$, where u'_i is the input signal returned by the supervisor for car i . By employing the model and estimation of the disturbances explained above, we obtain the following compensating input:

$$c_i(t, y_i) = -\frac{g_i e^{-t/h_i} u'_i + d_{path,i}(y_i)}{g_i e^{-t/h_i} + f_i}.$$

This, in turn, simplifies (29) as $\dot{y}_i = v_i + d_{proj,i}$ and $\dot{v}_i = a_i v_i + b_i + f_i u'_i$. Eventually with the compensation, we consider the following car dynamics in the experiment. For $i \in \mathcal{C} \cup \bar{\mathcal{C}}$,

$$\begin{aligned}\dot{y}_i &= v_i + d_{y,i}, \\ \dot{v}_i &= \begin{cases} \max(0, a_i v_i + b_i + f_i u'_i + d_{v,i}) & \text{if } (v_i = v_{i,min}), \\ \min(0, a_i v_i + b_i + f_i u'_i + d_{v,i}) & \text{if } (v_i = v_{i,max}), \\ a_i v_i + b_i + f_i u'_i + d_{v,i} & \text{otherwise,} \end{cases}\end{aligned}$$

where $d_{v,i}$ represents an error from the compensation and an unpredicted source of disturbance.

7.3 Results

The on-board computers on cars 1 and 2 run Algorithm 2. Since the same algorithm is run with the same measurement updated every $\tau = 0.1s$, the outputs are the same, thereby working as centralized control.

The parameters are as follows: $a = (-0.53, -0.30, -0.43) s^{-1}$, $b = (-84.68, -66.43, -49.64) cm/s^2$, $g = (0.99, 0.44, 0.95)$, and $h = (9.17, 6.84, 5.95) s$. The gain $f_i(t)$ for all i is a time-varying parameter, estimated at every time step. The bounds of the speed are $v_{min} = (10.5, 10.5, 13) cm/s$ and $v_{max} = (17, 16.5, 15) cm/s$, and those of the motor input (PWM) are $u_{min} = (105, 105, 130)$ and $u_{max} = (170, 165, 150)$. The bounds of the measurement noises are empirically chosen by ignoring long tails as $\delta_{y,i,min} = -25 cm$, $\delta_{y,i,max} = 25 cm$, $\delta_{v,i,min} = -25 cm/s$, and $\delta_{v,i,max} = 16 cm/s$ for all i . The bounds of the experimental disturbance are $d_{y,min} = (-5, -3, -3) cm/s$, $d_{y,max} = (3, 4, 3) cm/s$, $d_{v,min} = (-4, -2, -3) cm/s^2$, and $d_{v,max} = (2, 3, 2.5) cm/s^2$.

Figure 11 depicts two experimental results obtained by implementing the exact supervisor. The intersection is located at $(0, 65) cm$ for all cars. The intersection is an area containing the point at which the paths intersect, as in Figure 8b.

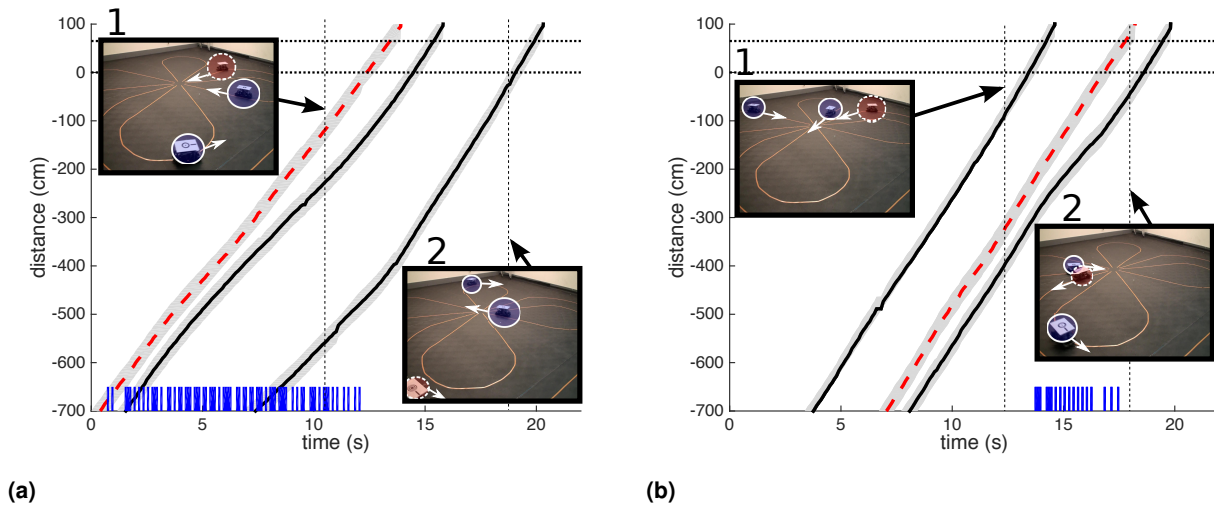


Figure 11. The results of implementing the exact supervisor (see also Extension 1). The intersection is represented by the dotted lines located at $(0, 65)$ cm. The solid black lines represent the position measurement of the controlled cars, cars 1 and 2, and the dotted red line represents the position measurement of the uncontrolled car, car 3. These figures demonstrate that the cars are not inside the intersection at the same time.

In Figure 11a, in picture 1, the uncontrolled car (dotted red circle) approaches the intersection earlier than the other cars. The supervisor overrides the controlled cars (solid blue circles) to decelerate them until their desired inputs do not cause conflict. In picture 2, the conflict is resolved, and one controlled car crosses the intersection alone without overrides. In Figure 11b, in picture 1, one controlled car approaches the intersection first. The supervisor lets this car accelerate and the other controlled car decelerate so that the uncontrolled car safely crosses the intersection, as shown in picture 2. Notice that the upper bound of the last car's position enters the intersection right after the lower bound of the uncontrolled car's position has exited, indicating that the override was applied because it is deemed necessary to avoid the collision. In both cases, intersection collisions are averted.

Figure 12 depicts 509 trajectories (semitransparent black lines) near the Bad set (red blocks). The trajectories are indicated in blue when the supervisor intervenes. From the 2D projections (figures on the right-hand side), we can confirm that none of the trajectories enters the Bad set. Since cars 1 and 2 are programmed to maintain constant speeds and car 3 does not change its speed quickly, the trajectories in the projections should be straight lines without overrides. We can see that the supervisor overrides cars 1 and 2 when the trajectories would enter the Bad set if the trajectories were linear. We observed in 15 trajectories that the cars were forced to stop before the Bad set because the supervisor could not find a safe input. This was because we had to truncate the tails of the distributions of disturbances, and thus the state measurement and the state estimation can

sometimes be incompatible. This truncation is necessary in the confined laboratory because otherwise a feasible initial condition may not always exist. The existence of a feasible initial condition is a necessary condition to initiate procedure SUPERVISOR in Algorithm 2.

8 Conclusions

We have designed exact and efficient supervisors that override controlled vehicles when collisions are imminent. The sources of uncertainty, such as measurement errors, unmodeled dynamics, and the presence of uncontrolled vehicles are taken into account in the design of the supervisors. The exact supervisor determines the existence of safe inputs (verification problem) by solving the Inserted Idle-Time (IIT) scheduling problem, which is proven to yield equivalent answers to the verification problem. To address the computational complexity issue, we also design the efficient supervisor that solves the IIT scheduling problem with a quantified approximation bound. The simulation results show that the efficient supervisor prevents collisions without substantial conservatism, compared to the exact one. The experiment using RC cars on an intersection testbed validated that collisions at an intersection are successfully averted by the exact supervisor.

Although this paper deals only with decision problems to focus on safety, there is no barrier to incorporate objective functions to address other issues such as fuel consumption or traffic congestion. The intersection considered in this paper is modeled as a single conflict area so that vehicles are required to occupy the intersection one at a time. This

assumption may make the system very conservative in that, for example, two vehicles turning right on different lanes are geometrically unable to collide while the supervisors do not let them inside the intersection at the same time. We are currently investigating the design of supervisors with a more general intersection model, which includes multi-conflict points. The result with simple first-order vehicle dynamics can be found in Ahn and Del Vecchio (2016, To appear). Other remaining issues include preventing rear-end collisions (Colombo and Del Vecchio (2014)) and considering unknown routes of vehicles.

Funding

This work was in part supported by NSF Award #1239182. Alessandro Colombo was in part supported by grant AD14VARI02 - Sottomisura B.

References

- Ahn H, Colombo A and Del Vecchio D (2014) Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles. In: *American Control Conference (ACC)*. pp. 867–873.
- Ahn H and Del Vecchio D (2016, To appear) Semi-autonomous intersection collision avoidance through job-shop scheduling. In: *the 19th ACM international conference on Hybrid Systems: Computation and Control*.
- Ahn H, Rizzi A, Colombo A and Del Vecchio D (2015) Experimental testing of a semi-autonomous multi-vehicle collision avoidance algorithm at an intersection testbed. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 4834–4839.
- Alonso-Ayuso A, Escudero L and Martn-Campo F (2011) Collision Avoidance in Air Traffic Management: A Mixed-Integer Linear Optimization Approach. *IEEE Transactions on Intelligent Transportation Systems* 12(1): 47–57.
- Borrelli F, Subramanian D, Raghunathan A and Biegler L (2006) MILP and NLP Techniques for centralized trajectory planning of multiple unmanned air vehicles. In: *American Control Conference*.
- Bruni L, Colombo A and Del Vecchio D (2013) Robust multi-agent collision avoidance through scheduling. In: *IEEE Conference on Decision and Control (CDC)*. pp. 3944–3950.
- Cassandras CG and Lafortune S (eds.) (2008) *Introduction to Discrete Event Systems*. Boston, MA: Springer US.
- Colombo A and Del Vecchio D (2012) Efficient algorithms for collision avoidance at intersections. In: *the 15th ACM international conference on Hybrid Systems: Computation and Control*. pp. 145–154.
- Colombo A and Del Vecchio D (2014) Least Restrictive Supervisors for Intersection Collision Avoidance: A Scheduling Approach. *IEEE Transactions on Automatic Control* DOI: 10.1109/TAC.2014.2381453.
- Cormen TH, Leiserson CE, Rivest RL and Stein C (2009) *Introduction to Algorithms*. 3rd edition edition. Cambridge: The MIT Press. ISBN 9780262033848.
- Garey MR, Johnson DS, Simons BB and Tarjan RE (1981) Scheduling UnitTime tasks with arbitrary release times and deadlines. *SIAM Journal on Computing* 10: 256–269.
- Gillula JH, Hoffmann GM, Huang H, Vitus MP and Tomlin CJ (2011) Applications of hybrid reachability analysis to robotic aerial vehicles. *The International Journal of Robotics Research* 30(3): 335–354.
- Hafner M, Cunningham D, Caminiti L and Del Vecchio D (2013) Cooperative Collision Avoidance at Intersections: Algorithms and Experiments. *IEEE Transactions on Intelligent Transportation Systems* 14(3): 1162–1175.
- Hafner MR and Del Vecchio D (2011) Computational tools for the safety control of a class of piecewise continuous systems with imperfect information on a partial order. *SIAM Journal on Control and Optimization* 49: 2463–2493.
- Hoffmann G and Tomlin C (2008) Decentralized cooperative collision avoidance for acceleration constrained vehicles. In: *IEEE Conference on Decision and Control (CDC)*. pp. 4357–4363.
- Kamal M, Imura J, Hayakawa T, Ohata A and Aihara K (2014) A Vehicle-Intersection Coordination Scheme for Smooth Flows of Traffic Without Using Traffic Lights. *IEEE Transactions on Intelligent Transportation Systems* DOI:10.1109/TITS.2014.2354380.
- Kowshik H, Caveney D and Kumar P (2011) Provable systemwide safety in intelligent intersections. *IEEE Transactions on Vehicular Technology* 60: 804–818.
- Lee J and Park B (2012) Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Transactions on Intelligent Transportation Systems* 13: 81–90.
- Maimone M, Cheng Y and Matthies L (2007) Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics* 24(3): 169–186. DOI:10.1002/rob.20184.
- Mastellone S, Stipanovi DM, Graunke CR, Intlekofer KA and Spong MW (2008) Formation Control and Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments. *The International Journal of Robotics Research* 27(1): 107–126.
- Pallottino L, Feron E and Bicchi A (2002) Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems* 3(1): 3–11.
- Peng J and Akella S (2005a) Coordinating Multiple Double Integrator Robots on a Roadmap: Convexity and Global Optimality. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 2751–2758.
- Peng J and Akella S (2005b) Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths. *The*

- International Journal of Robotics Research* 24(4): 295–310.
- Richards A, Schouwenaars T, How JP and Feron E (2002) Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance, Control, and Dynamics* 25(4): 755–764.
- Rizzi A (2014) Analysis and optimization of an experimental apparatus to test active safety systems in vehicles. URL <https://www.politesi.polimi.it/handle/10589/92232>.
- Smith RN, Chao Y, Li PP, Caron DA, Jones BH and Sukhatme GS (2010) Planning and Implementing Trajectories for Autonomous Underwater Vehicles to Track Evolving Ocean Processes Based on Predictions from a Regional Ocean Model. *The International Journal of Robotics Research* 29(12): 1475–1497.
- Verma R, Del Vecchio D and Fathy H (2008) Development of a Scaled Vehicle With Longitudinal Dynamics of an HMMWV for an ITS Testbed. *IEEE/ASME Transactions on Mechatronics* 13(1): 46–57. DOI:10.1109/TMECH.2008.915820.
- Wu J, Abbas-Turki A and ElMoudni A (2012) Cooperative driving: an ant colony system for autonomous intersection management. *Applied Intelligence* 37: 207–222.
- Wurman PR, D’Andrea R and Mountz M (2008) Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine* 29(1): 9–19.

Appendix A: Index to Multimedia Extensions

Extension	Media type	Description
1	Video	This video contains experiments of the exact supervisor presented in Section 7.

Appendix B

We provide the proofs of Lemmas 1-3 in this section.

Lemma 1. *Procedure POLYNOMIAL in Algorithm 3 solves the IIT scheduling problem with unit process times and finds a feasible schedule if exists.*

Proof. To account for inserted idle-times, we define an initial set of forbidden regions as $F_{0,\gamma} = (\bar{r}_\gamma - 1, \bar{p}_\gamma)$ for all γ . If $t_j \notin F_{0,\gamma}$ for some j , then we have $(t_j, t_j + 1) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset$, thereby satisfying condition (21).

Forbidden Region Declaration in lines 3-11 of procedure POLYNOMIAL solves Problem 4. The key idea is critical time c , which is the latest start time of job σ_i . If job σ_i starts later than critical time c , at least one of the subsequent jobs cannot be scheduled before its deadline. That is, if a job cannot start before the critical time (line 10), there is no schedule satisfying conditions (19)-(21). Otherwise, there will be a

schedule satisfying all the conditions. This relation between critical time and the existence of a feasible schedule was proved in Garey et al. (1981) with initially empty forbidden regions. Since initially non-empty forbidden regions do not affect the fact that critical time is the latest start time, this analysis of critical time can determine the existence of a schedule with non-empty initial forbidden regions.

With the forbidden regions declared, the EDD rule finds a schedule in lines 12-17. Time s is assigned to each job as a schedule. In line 14, s avoids the forbidden regions so that condition (21) is satisfied. If there is no ready job (line 15), a job with the minimum release time among unscheduled jobs is scheduled, and otherwise (line 16), a job with the earliest deadline among ready jobs is scheduled. After a job is scheduled, line 17 updates s to $s + 1$ so that condition (20) is satisfied. This schedule cannot satisfy condition (19) if ans has been *no* in line 10. Otherwise, this schedule satisfies conditions (19)-(21). \square

Lemma 2. *If $\text{APPROX}([x^a(0), x^b(0)], S) = (\mathbf{T}, \text{yes})$, and $\text{RELAXEDEXACT}([x^a(0), x^b(0)], S) = (\bar{\mathbf{T}}, \pi^*, \text{yes})$, then $T_j \leq \bar{T}_j$ for all $j \in \mathcal{C}$.*

Proof. We will show by induction on j that $T_{\pi_j^*} \leq \bar{T}_{\pi_j^*}$ for all $\pi_j^* \in \mathcal{M}$. Notice that $T_i = \bar{T}_i = 0$ for all $i \in \bar{\mathcal{M}}$. The schedule \mathbf{T} is generated by procedure SCHEDULING in Algorithm 1. For the base case, $T_{\pi_1^*} = \max(R_{\pi_1^*}, P_{max})$ in line 8 of Algorithm 1. Since $\bar{T}_{\pi_1^*}$ is a feasible solution of Problem 5 by Lemma 1, it satisfies $R_{\pi_1^*} \leq \bar{T}_{\pi_1^*}$ from condition (23) and $(0, P_i(0)) \cap (\bar{T}_{\pi_1^*}, \bar{T}_{\pi_1^*} + \theta_{max}) = \emptyset$ for all $i \in \bar{\mathcal{M}}$ from condition (24). Thus, $\max(R_{\pi_1^*}, P_{max}) = T_{\pi_1^*} \leq \bar{T}_{\pi_1^*}$. Now, suppose $T_{\pi_{k-1}^*} \leq \bar{T}_{\pi_{k-1}^*}$. Then, for $j = k$, we need to show that $T_{\pi_k^*} \leq \bar{T}_{\pi_k^*}$.

In line 10 of Algorithm 1, $T_{\pi_k^*} = \max(R_{\pi_k^*}, T_{\pi_{k-1}^*} + P_{\pi_{k-1}^*}(T_{\pi_{k-1}^*}))$. If $T_{\pi_k^*} = R_{\pi_k^*}$, we have $T_{\pi_k^*} \leq \bar{T}_{\pi_k^*}$ because $\bar{T}_{\pi_k^*}$ satisfies condition (23). If $T_{\pi_k^*} = T_{\pi_{k-1}^*} + P_{\pi_{k-1}^*}(T_{\pi_{k-1}^*})$, we have $T_{\pi_k^*} \leq \bar{T}_{\pi_{k-1}^*} + \theta_{max}$ because $T_{\pi_{k-1}^*} \leq \bar{T}_{\pi_{k-1}^*}$ and $P_{\pi_{k-1}^*}(T_{\pi_{k-1}^*}) \leq \theta_{max}$ by (22). Since $\bar{T}_{\pi_k^*}$ satisfies condition (24), $\bar{T}_{\pi_{k-1}^*} + \theta_{max} \leq \bar{T}_{\pi_k^*}$. Therefore, $T_{\pi_k^*} \leq \bar{T}_{\pi_k^*}$. In lines 13 and 15 of Algorithm 1, the schedule can increase so that $T_{\pi_k^*} = \bar{P}_\gamma$ for some $\gamma \in \bar{\mathcal{C}}$ if $T_{\pi_k^*} \geq \bar{R}_\gamma$ or $T_{\pi_k^*} + P_{\pi_k^*}(T_{\pi_k^*}) > \bar{R}_\gamma$. Since $\bar{T}_{\pi_k^*}$ satisfies condition (25), if $\bar{T}_{\pi_k^*} \geq \bar{R}_\gamma$ or $\bar{T}_{\pi_k^*} + \theta_{max} > \bar{R}_\gamma$, then it must be $\bar{T}_{\pi_k^*} \geq \bar{P}_\gamma$. Therefore, in either case, $T_{\pi_k^*} \leq \bar{T}_{\pi_k^*}$. \square

Lemma 3. *If $\text{APPROX}([x^a(0), x^b(0)], S) = (\emptyset, \text{no})$, then $\text{RELAXEDEXACT}([x^a(0), x^b(0)], S) = (\emptyset, \pi^*, \text{no})$.*

Proof. In Algorithm 5, procedure APPROX returns *no* if $[y^a(0), y^b(0)] \cap B \neq \emptyset$ in line 2 or procedure SCHEDULING with π^* returns *no* in line 5. In the former case, procedure RELAXEDEXACT also returns *no* as in line 2

of Algorithm 4. The latter case implies that $T_{\pi_j^*} > D_{\pi_j^*}$ for some $\pi_j^* \in \mathcal{C}$. By Lemma 2, we have $\bar{T}_{\pi_j^*} \geq T_{\pi_j^*}$ if a feasible solution $\bar{\mathbf{T}}$ of Problem 5 exists. Such a schedule cannot be feasible because $\bar{T}_{\pi_j^*} \geq T_{\pi_j^*} > D_{\pi_j^*}$, which does not satisfy condition (23). Thus, procedure RELAXEDEXACT returns (\emptyset, π^*, no) . \square

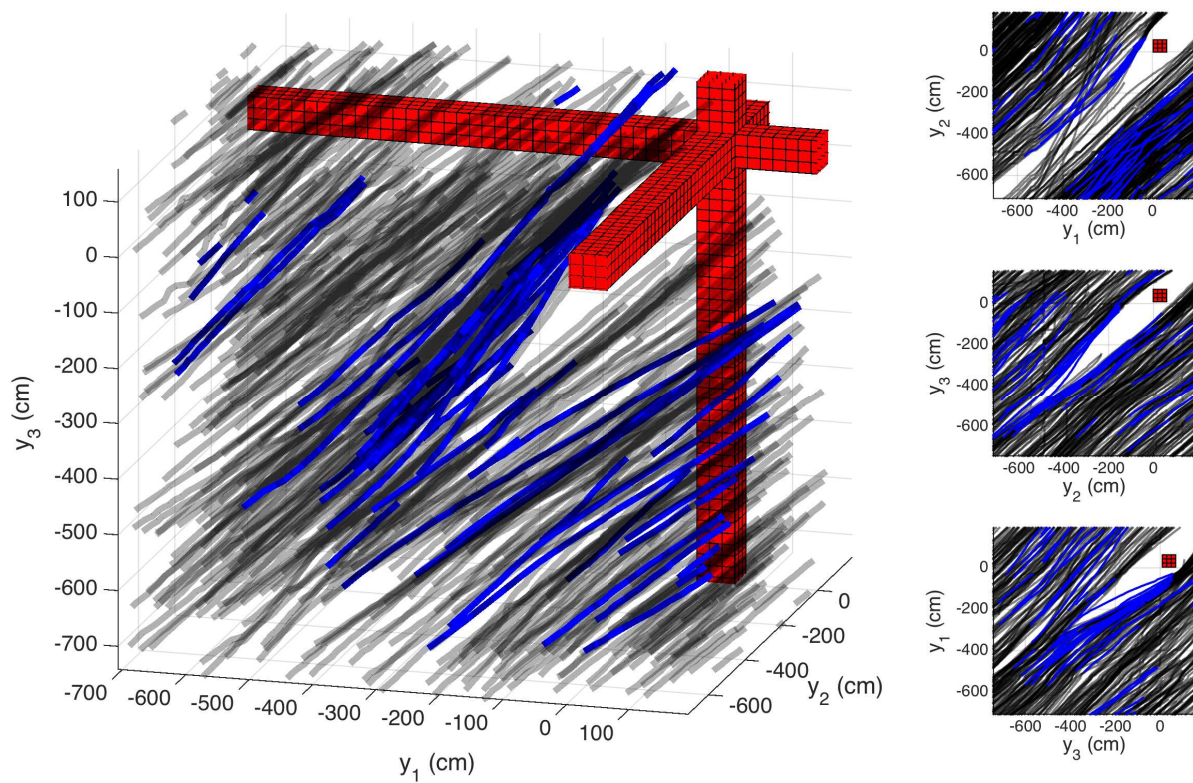


Figure 12. Trajectories (semitransparent black lines) in the output space. There are 509 trajectories, which are indicated in blue when the supervisor intervenes. The three figures on the right-hand side are the 2D projections. These confirm that all of the trajectories avoid the Bad set.